

**REPRODUCIBLE COPY
(FACILITY CASEFILE COPY)**

NASA CR-145024

**AIRBORNE ADVANCED
RECONFIGURABLE COMPUTER SYSTEM
(ARCS)**

B. E. Bjurman, G. M. Jenkins, C. J. Masreliez, K. L. McClellan, and J. E. Templeman

(NASA-CR-145024) AIRBORNE ADVANCED
RECONFIGURABLE COMPUTER SYSTEM (ARCS) Final
Report, Mar. 1975 - Apr. 1976 (Boeing
Commercial Airplane Co., Seattle) 547 p HC
\$13.00

N76-30865

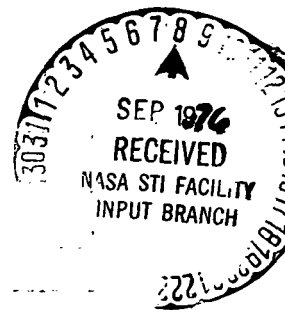
Unclas
48748

CSCL 09B G3/62

August 1976

Prepared under contract NAS1-13654 by
Boeing Commercial Airplane Company
P.O. Box 3707
Seattle, Washington 98124

for
Langley Research Center
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
Hampton, Virginia 23665



1. Report No. NASA CR-145024		2. Government Accession No.		3. Recipient's Catalog No.	
4. Title and Subtitle AIRBORNE ADVANCED RECONFIGURABLE COMPUTER SYSTEM (ARCS)				5. Report Date August 1976	
				6. Performing Organization Code	
7. Author(s) B. E. Bjurman, G. M. Jenkins, C. J. Masreliez, K. L. McClellan, and J. E. Templeman				8. Performing Organization Report No. D6-42476	
9. Performing Organization Name and Address Boeing Commercial Airplane Company P.O. Box 3707 Seattle, Washington 98124				10. Work Unit No.	
				11. Contract or Grant No. NAS1-13654	
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Langley Research Center Hampton, Virginia 23665				13. Type of Report and Period Covered Final Report - March 1975 through April 1976	
				14. Sponsoring Agency Code	
15. Supplementary Notes ARCS technical monitor: Mr. A. O. Lupton FID/NASA LRC					
16. Abstract <p>Fault-tolerant electronic subsystems are being applied to future commercial transports to improve their range-payload performance and to enhance their operational capability/reliability. This 1-year study defined a new digital computer subsystem fault-tolerant concept and assessed the potential benefits and costs of such a subsystem when used as the central element of a new transport's flight control system. The derived Advanced Reconfigurable Computer System (ARCS) is a triple-redundant computer subsystem that automatically reconfigures, under multiple fault conditions, from triplex to duplex to simplex operation, with redundancy recovery if the fault condition is transient. The study included criteria development covering factors at the aircraft's operational level that would influence the design of a fault-tolerant system for commercial airline use.</p> <p>A new reliability analysis tool was developed for evaluating redundant, fault-tolerant system availability and survivability; and a stringent digital system software design methodology was used to achieve design/implementation visibility.</p>					
17. Key Words (Suggested by Author(s)) Airborne computer Redundant Avionics Reliability analysis Fault tolerant Software design Reconfigurable				18. Distribution Statement	
19. Security Classif. (of this report) Unclassified		20. Security Classif. (of this page) Unclassified		21. No. of Pages 535	
				22. Price*	

*For sale by the National Technical Information Service, Springfield, Virginia 22151

CONTENTS

1.0	SUMMARY	1
2.0	INTRODUCTION	4
2.1	Program Overview	4
2.2	Statement of Work Summary and TCV Program Relationship	6
2.3	Report Organization	9
2.4	Acknowledgment	9
3.0	SYMBOLS AND ABBREVIATIONS	10
4.0	DESIGN CRITERIA	14
4.1	Operational Criteria	14
4.1.1	Economic Considerations	16
4.1.2	Flight Safety Considerations	36
4.2	Arcs Design Requirements	43
4.2.1	System Functional Design	43
4.2.2	Software Design	50
4.2.3	Hardware Design	50
5.0	ARCS DESIGN CONCEPT	55
5.1	Functional Description	55
5.1.1	System Reconfiguration	55
5.1.2	Functional Organization	68
5.1.3	Sensor Signal Selection and Fault Detection (SSFD)	75
5.1.4	System Self-Test	79
5.2	Software Design	82
5.2.1	Top-Down Design (Step 1)	83
5.2.2	Software Design Tree (Step 2)	83
5.2.3	Module Identification (Step 3)	85
5.2.4	Transition Diagram (Step 4)	86
5.2.5	Software Code (Step 5)	89
5.3	Hardware Design	89
5.3.1	Hardware System Architecture	90
5.3.2	Hardware System Interfaces	94
5.3.3	Arcs Computer Unit	100
5.3.4	System Test Panel	116
6.0	ARCS DESIGN ANALYSIS	
6.1	Fault-Tolerance Analysis	120
6.1.1	Fault Analysis	120
6.1.2	Reliability Analysis	131
6.2	Cost/Benefit Analysis	169
6.2.1	ARCS Availability Cost Effect	171
6.2.2	Acquisition Cost	174

CONTENTS (Concluded)

6.2.3	System Test Cost Effect	174
6.2.4	Cost/Benefit Summary	186
7.0	ARCS IMPLEMENTATION	
7.1	ARCS Design Specification	188
7.1.1	ARCS Real-Time Operations	188
7.1.2	Ground Test Operations	190
7.1.3	Asynchronous Operations	191
7.1.4	Redundancy Management	191
7.1.5	Hardware Design Requirements	193
7.2	Feasibility of WWCS Modification	194
7.2.1	WWCS-ARCS Comparison	195
7.2.2	WWCS Modifications	195
8.0	CONCLUDING SECTION	197
8.1	Summary of Results	197
8.1.1	ARCS Design Criteria	197
8.1.2	ARCS Design Concept	198
8.1.3	ARCS Design Analysis	199
8.2	Conclusions	201
APPENDIX A		203
APPENDIX B		217
APPENDIX C		311
APPENDIX D		314
APPENDIX E		359
APPENDIX F		396
APPENDIX G		437
APPENDIX H		479
APPENDIX I		519
REFERENCES		535

TABLES

No.		Page
1	Power-On/Power-Fault Recovery Strategy	62
2	Functional Characteristics Summary of ARCS Processor	102
3	Recovery Strategy	125
4	Surveyed Reliability Programs and Associated Analysis Methods	133
5	Reliability Program Features	133
6	Near- and Intermediate-Term CWS Failure Probabilities	150
7	Probability of No Module Failure	152
8	Autoland Availability/Failure Probabability Results for ARCS Near-Term Baseline	154
9	Autoland Availability/Failure Probability Results for Intermediate-Term Baseline	154
10	ARCS Baseline Far-Term Results	155
11	ARCS Voting Node Trade Study Results for the CWS Mode Application	158
12	Definition of Stage Numbers for the ARCS Near-Term Application	161
13	Two-Step Failure Transitions Leading to System Failure for the ARCS Near-Term CWS Mode Application	162
14	ARCS Sensitivities for the Near-Term Application Model	164
15	Contribution of Computer Unreliability to System Failure Probability	165
16	Weather-Caused Interruptions	173
17	Line Maintenance Activity With System Test Capability	177
18	Line Maintenance Activity Without System Test Capability	177
19	Shop Maintenance Operation	180
20	Sensor System Removals	183
21	Sensor Maintenance Parameters	183
22	Maintenance Costs Per Year	186
23	Processing and Interface Requirements for ARCS	189
B1	Software Index	229
B2	Channel Identification	245
B3	Left and Right Channel Identification	245
B4	Monitor Flag Meaning	261
B5	Output Monitors	263
B6	Monitor Trip Counters	264
B7	Permanent Failure Flags	268
E1	Simulated Hardware Indicators	360
E2	Synchronization Indicators	361
E3	Recovery Indicators for Computer A	363
E4	Recovery Indicators for Computer B	364
E5	Recovery Indicators for Computer C	365
E6	Do Not Use, Permanent Flags and Counters for A, B, and C Signals for Computer A	366
E7	Do Not Use, Permanent Flags and Counters for A, B, and C Signals for Computer B	367

TABLES (Continued)

No.	Page
E8 Do Not Use, Permanent Flags and Counters for A, B, and C Signals for Computer C	368
E9 Input and Output for Computer A	369
E10 Input and Output for Computer B	370
E11 Input and Output for Computer C	371
E12 Simulated Hardware Indicators	373
E13 Synchronization Indicators	374
E14 Recovery Indicators for Computer A	375
E15 Recovery Indicators for Computer B	376
E16 Recovery Indicators for Computer C	377
E17 Do Not Use, Permanent Flags and Counters for A, B, and C Signals for Computer A	378
E18 Do Not Use, Permanent Flags and Counters for A, B, and C Signals for Computer B	379
E19 Do Not Use, Permanent Flags and Counters for A, B, and C Signals for Computer C	380
E20 Input and Output for Computer A	381
E21 Input and Output for Computer B	382
E22 Input and Output for Computer C	383
E23 Simulated Hardware Indicators	385
E24 Synchronization Indicators	386
E25 Recovery Indicators for Computer A	387
E26 Recovery Indicators for Computer B	388
E27 Recovery Indicators for Computer C	389
E28 Do Not Use, Permanent Flags and Counters for A, B, and C Signals for Computer A	390
E29 Do Not Use, Permanent Flags and Counters for A, B, and C Signals for Computer B	391
E30 Do Not Use, Permanent Flags and Counters for A, B, and C Signals for Computer C	392
E31 Input and Output for Computer A	393
E32 Input and Output for Computer B	394
E33 Input and Output for Computer C	395
F1 CARSRA Input Data	421
F2 CARSRA Example Transition Rates/10 ⁶ Hrs	426
F3 CARSRA Input Example	427
F4 CARSRA Output Example	429
H1 Sources Utilized for the MARKOV Model Parameter Assessment Task	480
H2 Permanent Fault Rates	481
H3 Normal Power Transients (Examples)	483
H4 Abnormal Power Transients (Examples)	483
H5 Flight Control Sensor Difference Standard Deviations	500
H6 ICPS SS/FD Thresholds and Time Delays	503
H7 Predicted ICPS Transient Sensor Failure Data in Turbulent Flight	506

TABLES (Concluded)

No.		Page
H8	Sensor Biases and Scale Factors	507
H9	ARCS Predicted Transient Sensor Failure Data in Turbulence	511
H10	Assessed Sensor Rate Ratios	517

FIGURES (Continued)

No.		Page
44	Watchdog Monitor Timing Diagram	112
45	Power Supply Block Diagram	114
46	System Test Panel Block Diagram	117
47	ARCS System Test Panel Layout	118
48	ARCS Synthesis/Analysis Scope	121
49	ARCS/WWCS Fault-Tolerance Analysis Overview	122
50	Local Computer's Assessment of a System Function	126
51	Power-On/Watchdog Monitor Trip Recovery as Processed by an Operating Computer (Triplex Operation Attained)	127
52	Power-On and Watchdog Monitor Trip Recovery as Processed by an Operating Computer (Duplex Operation Attained)	128
53	Power-On/Watchdog Monitor Trip Recovery as Processed by an Operating Computer (Simplex Operation Attained)	129
54	Power-On/Watchdog Monitor Trip as Processed by an Operating Computer (Triplex-to-Duplex Degradation)	130
55	Example of Markov Model of a Triplex Stage	135
56	Flight Control System Dependency Tree	136
57	Functional Readiness Variations During Scheduled Maintenance	138
58	Reliability Study Components	140
59	Near-Term ARCS Dependency Tree	141
60	Intermediate-Term ARCS Dependency Tree	142
61	Far-Term ARCS Dependency Tree	143
62	Near-Term/Intermediate-Term Markov Stage Model	145
63	Far-Term Markov Stage Model	147
64	Voting Trade Study Configurations	157
65	Latent Sensor Failure Illustration	162
66	Latent Sensor Failure Model	167
67	Avionics Cost Factors	170
68	Line Maintenance Operation	176
69	Shop Maintenance Operation	179
A1	Application Control Law Overview	204
A2	Roll Autoland Control Law	205
A3	Pitch Autoland Control Law	206
A4	Yaw Autoland Control Law	207
A5	Go-Around Control Law	208
A6	Pitch Command Augmentation Control Law (With γ Hold)	208
A7	Roll Command Augmentation Control Law	209
A8	Yaw Command Augmentation Control Law	209
A9	Maneuver and Gust Load Alleviation Control Law	210
A10	Flutter Suppression Control Law	210
A11	Sensor and Mode Control Block—Near Term Application Model	211
A12	Servo and Display Block—Near Term Application Model	212
A13	Sensor and Mode Control Block—Intermediate Term Application Model	213
A14	Servo and Display Block Intermediate Term Application Model	214
A15	Sensor and Mode Control Block—Far Term Application Model	215
A16	Servo and Display Block—Far Term Application Block	216

FIGURES (Continued)

No.		Page
B1-A	ARCS Functional Tree	218
B1-B	Ground Test Functional Requirements	219
B1-C	Background Tasks Requirements Tree	220
B1-D	ARCS Redundancy Management Functional Tree	221
B2-A	ARCS Software Structure	222
B2-B	ARCS Software Structure	223
B2-C	ARCS Software Structure	224
B2-D	ARCS Software Structure Tree	225
B3-A	ARCS Transition Diagram	226
B3-B	Redundancy Management Transition Diagram	227
B4	Scheduling Table	228
B5	Tree Breakdown of System Status	241
B6	Update Function	252
B7	Continuous Signal Selection/Fault Detection Algorithm	258
B8	Triplex Monitoring	262
B9	Duplex Monitor	263
B10	Triplex Monitor Trip Assessment	265
B11	Duplex Monitor Trip Assessment	266
B12	General Failure Status Assessment	267
B13	Servo Management Functional Breakdown	271
B14	Control Law Tree	279
B15	Inputs and Outputs of Control Laws	280
B16	Ground Test Interrupt Processing	287
B17	Maintenance Test Functional Requirements	289
B18	Test Structure	293
B19	Servo Test Functional Breakdown	299
B20	Display Function	302
B21	Ground Test Display Operation	303
D1	MCP-701 Throughput Prediction	317
D2	Aerospace Processor Throughput Comparison	320
D3	Aerospace Processor Input/Output Overhead Burden	321
D4	Cross-Channel Word Format	329
D5	Cable Failure Examples	331
D6	Summary of Candidate Sync Routines	338
D7	Skew Definition	340
D8	The "Wait" Algorithm for Frame Synchronization	341
D9	The "Master" Algorithm for Frame Synchronization	343
D10	"Time Window" Point of View for Failure Detection	344
D11	Failure Detection Based Exclusively on Wait Times	346
D12	Additional Failures Detected With Test For Sync Indicator Clear at Beginning of Routine	348
D13	Identical Failures Detected in All Computers For Master Approach	349
D14	Candidate Triplex Actuator For ARCS	353
F1	Example of Stage Markov Model	398
F2	Flight Control System Dependency Tree	400

FIGURES (Continued)

No.		Page
F3	401
F4	401
F5	CARSRA Structure	405
F6	MAIN Program Flow Diagram	407
F7	READIN Flow Diagram	408
F8	INITYZ Flow Diagram	409
F9	FORMT Flow Diagram	410
F10	COMPUTE Flow Diagram	412
F11	Subroutine AVAIL Flow Diagram	413
F12	Subroutine SETETY Flow	415
F13	Flow Diagram For Subroutine FAILPR	417
F14	INDFP Flow Diagram	418
F15	DEFP Flow Diagram	419
F16	424
F17	Triplex Stage MARKOV Model	425
F18	Duplex Stage MARKOV Model	425
G1	GE's Coverage Study Plan—TASK OUTLINE	438
G2	Markov Model Concepts Necessary for Coverage Definition	440
G3	Simple Triplex Stage Model	442
G4	446
G5	448
G6	454
G7	455
G8	Lower Limits for (N/M) at 60% and 90% Confidence Levels	459
G9	ARCS Redundancy Management Block Diagram	461
G11	Laboratory Breadboard Block Diagram	462
G12	463
G13	Cross-Channel Coil Sum Current Monitor	464
G14	A/D—D/A Self-Test Loop	465
G15	Watchdog Monitor	466
G16	Breadboard Servo Electronics Schematic	467
G17	Breadboard Demonstration and Analysis Objectives	468
G18	Groundrules for FMEA and Fault Table	469
G19	Failure Modes Considered in Fault Table	470
G20	Outline of the Failure Modes and Effects Analysis (FMEA) Method	471
G21	Typical Form Used to Construct Fault Table and Perform FMEA	472
G22	Outline of Statistical Estimation Method	473
G23	Results of FMEA for Duplex-to-Simplex Transition Case	474
G24	Data From Random Sample Fault Insertion for Duplex-to-Simplex Transition Case	475
G25	Overall Results for Servo Stage Second Failure Coverage Evaluation	476
H1	Characteristic Waveform of Lightning Stroke	485
H2	Characteristic Electrical Transient Induced by Lightning	485
H3	Sensor Redundancy Management Strategy at Triple (or Quad) Redundancy	488
H4	490

FIGURES (Concluded)

No.		Page
H5	Latent Failure Situation	491
H6	Sensor Redundancy Management at Duplex Redundancy	495
H7	ICPS SSFD Algorithm	501
H8	Analytically Derived Altimeter Signal	515
I1	Cross Channel Link Configurations	522
I2	Triplicated Tee Configuration	524
I3	Cross Channel Data Link Hardware—Block Diagram	532

AIRBORNE ADVANCED RECONFIGURABLE COMPUTER SYSTEM (ARCS) DEFINITION STUDY

B. E. Bjurman, G. M. Jenkins, C. J. Masreliez, K. L. McClellan, J. E. Templeman
Boeing Commercial Airplane Company

1.0 SUMMARY

The Airborne Advanced Reconfigurable Computer System, or ARCS, program (NASA contract NAS1-13654) was a 1-year study to (1) define a new digital computing system functional concept employing contemporary ideas of fault tolerance, (2) substantiate system performance improvements through the use of advanced reliability assessment methods, and (3) apply methods for assessing potential benefits and costs of fault-tolerant computer technology, as applied to future commercial transport avionics. The study was sponsored by NASA Langley Research Center under the Terminal Configured Vehicle (TCV) program. Boeing Commercial Airplane Company was the prime contractor, with General Electric, United Airlines, and Boeing Computer Services as subcontractors.

The ARCS represents a unique combination of airline, aircraft manufacturer, and avionic systems manufacturer participation in formulating a new fault-tolerant airborne computer system architecture. Comparison of the resulting conceptual design with contemporary system technology indicates improved airline operator profitability. This is a consequence of enhanced system availability, lowered cost of replicated elements, and improved system maintainability.

Other significant ARCS program results were the development of a reliability analysis tool for evaluating redundant fault-tolerant systems and the adaptation of advanced software design principles to define a stringent methodology for designing software for fault-tolerant systems. The important element of the software design methodology is the design visibility given to all parties involved in the design process.

Three primary tasks were involved in formulating the ARCS concept: criteria development, design synthesis, and design evaluation.

Operational considerations, such as airline economics and flight safety, associated with an autoland function and a control wheel steering (CWS) type of stability/command augmentation function were used to define survivability and functional availability criteria for the ARCS. A system architecture with autonomous, redundant channels, capable of operating down to simplex without pilot intervention and expandable to quadruplex redundancy, was identified as a basic design requirement. An integrated test function to ensure system integrity and to enhance system maintenance was a further requirement.

In the developed design concept, system reconfiguration is managed by software processes driven by software and hardware monitors. The software was designed using top-down, structured programming principles to ensure maximum integrity and visibility. The single-box-per-channel hardware design uses a directly operable input/output concept with cross-channel communication performed via dedicated one-way optical data links. Sensors, mode controls, and servos are dedicated on a channel basis.

A fault analysis using a functional simulation demonstrated the validity of the reconfiguration processes down to the functional level from which circuit design and software implementation would begin. Survivability and availability of the ARCS design, including significant alternate design configurations, were assessed with the CARSRA reliability analysis program developed for the ARCS.

The analysis showed a fivefold improvement in survivability of the CWS function implemented in ARCS compared to a CWS function implemented with contemporary airborne computer technology (i.e. fail-operational/fail-passive capability in a triplex configuration), as represented by the baseline GE MCP 703 Whole Word Computer System (WWCS). A quadruplex ARCS with contemporary state-of-the-art component failure rates was shown to meet the survivability requirement for a flight-crucial fly-by-wire CWS control function, while a quadruple two-fail-operational/fail-passive system would not quite satisfy this requirement.

The survivability of the ARCS and WWCS autoland functions, when initially fault free, were both shown to well satisfy the requirement. The ARCS will also satisfy the specified autoland survivability criterion with one failure incurred before the initiation of the critical autoland phase. This results in an improved ARCS autoland function availability which means, in comparison with the WWCS, a factor of 4 reduction in diversions due to autoland unavailability for Category III operations. The greater functional survivability and availability results from reconfiguration to simplex. For some users, simplex may not be an acceptable mode of operation.

The cost/benefit analysis, using an airline model defined in the study, showed that the improved availability of the ARCS, the lower cost of system acquisition, and the potential maintenance cost reductions result in an annual saving in excess of \$4000 per ARCS-equipped aircraft compared to WWCS-equipped aircraft. Additional savings were identified but not quantified.

Low-visibility operation is not standard procedure by all airlines today. Category III autoland in particular is limited to a few airlines and airports worldwide. Low-visibility capability is not a dispatch requirement but an airline option. It was concluded that the full benefit of fault-tolerant computer technology will be reaped when the functions provided by the avionic systems become dispatch critical or otherwise mandated for use on every flight. Fully implemented active controls in future aircraft will require such functions.

Not all aspects of duplex-to-simplex reconfiguration were covered by the ARCS study; work remains in the areas of analytical redundancy (fault monitoring of simplex signals) and techniques for predicting and evaluating "coverage," i.e., the likelihood of function survival given a fault in the duplex state, with a high degree of confidence. Assessment of "coverage" is a central problem in designing fault-tolerant systems. Credible reliability estimations hinge on the level of confidence at which "coverage" values can be estimated for duplex to simplex operations. Application of an ARCS-developed method, based on random fault insertion, to a gate-level simulation/emulation of a candidate computer is seen as the next step in fault-tolerant technology consolidation.

The ARCS data supports application of fault-tolerant computer technology to increase the effectiveness of flight-critical avionics at lower cost to the operator. Fault-tolerant computing will be a significant element in achieving fly-by-wire and active controls technology, as well as general use of Cat III capability, at acceptable cost.

2.0 INTRODUCTION

The Airborne Advanced Reconfigurable Computer System, or ARCS, program (NASA contract NAS1-13654) was a 1-year study to (1) define a new digital computing system functional concept employing contemporary ideas of fault tolerance, (2) substantiate system performance improvements through the use of advanced reliability assessment methods, and (3) apply methods for assessing potential benefits and costs of fault-tolerant computer technology, as applied to future commercial transport avionics. The study was sponsored by NASA Langley Research Center under the Terminal Configured Vehicle (TCV) program. Boeing Commercial Airplane Company was the prime contractor, with General Electric, United Airlines, and Boeing Computer Services as subcontractors.

The ARCS program joined a team of airline, aircraft manufacturer, and avionic systems manufacturer personnel to formulate a new fault-tolerant airborne computer system architecture. The resulting conceptual design was compared with contemporary system technology to determine its impact on airline operator profitability. System availability, cost of replicated elements, and system maintainability were parameters in determining the new system's benefit/cost to an airline.

Other significant ARCS program elements were to identify or develop an appropriate reliability analysis tool for evaluating redundant fault-tolerant systems and to adapt advanced software design principles to define a stringent methodology for designing software for fault-tolerant systems. An important part of formulating the software design methodology was to identify those techniques that provide design visibility to all parties involved in the software design and implementation processes.

2.1 PROGRAM OVERVIEW

Three primary tasks were involved in formulating the ARCS system concept: criteria development, design synthesis, and design evaluation. Criteria development covered all factors on the aircraft operational level that influence the design of a fault-tolerant computer system. Of those, economic considerations address the projected functional scope of the system, the expected functional availability, and the required maintainability. Safety considerations yield criteria specifying the required survivability of critical functions, which for this study included controls-configured vehicle (CCV) fly-by-wire (FBW) functions and the Cat III autoland function.

The operational criteria were transformed into design requirements for a fault-tolerant (redundant) digital computer system, specifying wherever possible design principles that would obviate single-point system failures. The following fundamental design requirements were formulated for the ARCS triple-channel configuration.

- Each computer unit shall independently assess its, and the system's, operational status.
- No computer or combinations of computers shall interrupt another computer's normal operation.

- The system's redundant operation must start up, and recover from, transient fault conditions without flightcrew intervention.
- The system design must be able to achieve functional operation down to a simplex string of operable elements and be architecturally expandable to at least quadruplex redundancy.

The core of the ARCS functional concept, developed from the above requirements, is the reconfiguration processes of failure isolation, transient fault recovery, and redundancy degradation. The key to reconfiguration is fault detection, with the ARCS having five fault monitor functions that directly, upon activation, initiate action leading to reconfiguration. They are: power interrupt, sensor signal failure detection, computational output monitors, servo fault detectors, and a watchdog monitor. The watchdog monitor is mechanized in hardware to detect faults that interrupt the real-time processing of the computer. In addition to these primary, or first-level, monitors, second-level hardware failure detectors and hardware/software self-test functions provide fault localization when the system is operating at a duplex redundancy level.

In the flight control system configured around the ARCS, redundant channel processes are consolidated at two system voting nodes: at the sensor signal input to the control law computations and at the servoactuator output. The sensor signal selection process is mechanized in software, and the servo output voting node is a hydromechanical mechanization. Both are assumed to be designs with absolute integrity for first-failure detection.

The majority of the ARCS reconfiguration mechanisms are software processes, where it has been recognized that the crucial aspect of a fault-tolerant system's software is to ensure design integrity through design visibility. In the ARCS design, this was achieved through the strict application of top-down design principles and structured programming techniques, as well as the establishment of rigid documentation standards compatible with these concepts. Since the ARCS concept emphasizes software processes to achieve system fault-tolerant qualities, the hardware must be viewed as the vehicle to facilitate an effective software design.

The ARCS primary hardware element is a single computer unit, replicated on a per-channel basis to build up a triplex- or quadruplex-redundant fault-tolerant system. The computer unit contains a processor and all channel interface electronics. Sensor, mode control, and servo hardware interfaces are dedicated on a channel basis. All cross-channel communication is accomplished via dedicated one-way, serial, optical, digital data buses that independently interconnect each computer to each other computer, providing complete electrical isolation between channels. Each computer exclusively controls the engagement and shutdown of its own servos.

The ARCS concept was analyzed with respect to its fault-tolerant performance with two major objectives in mind: (1) to verify through fault analysis that the system concept in fact had the required fault-tolerant qualities from a functional point of view and (2) to assess, through a reliability analysis, the merit of those fault-tolerant qualities from a probabilistic point of view.

Simulation was used as a tool in the fault analysis because the reconfiguration processes, though each conceptually understandable, were difficult to validate using only a paper analysis. The simulation demonstrated that the ARCS fault-tolerant design concept was valid, even though the sensor signal selection/failure detection algorithms for the duplex-to-simplex degradation were not fully developed.

The reliability analysis work involved reviewing available reliability analysis methods and tools to identify those suitable for use in the ARCS study and assessing the reliability of the ARCS design and selected configuration alternatives using the related method and tool. Ten different reliability estimation programs were surveyed, six from Boeing and four from NASA. None could adequately handle the ARCS reliability assessment task; therefore, a new tool called CARSRA (Computer Aided Redundant System Reliability Analysis) was developed.

CARSRA was used to generate the projected reliability of the ARCS within the scope of the defined commercial transport operational applications. Assuming present-day hardware failure rates, a quadruplex ARCS-type system is required to meet the fly-by-wire reliability requirement, while a triplex ARCS will meet the Cat III autoland functional reliability with one module failed (sensor, computer, or servo) at the Cat III alert height. A comparison with contemporary redundant system design approaches showed that a fail-op/fail-op quadruplex system design would fall short of the fly-by-wire requirement, and a fail-op triplex system would have to have all modules working at the alert height to meet the Cat III requirement. The ARCS, with its ability to meet the reliability requirement for the Cat III case with one module failed, reduces by a factor of 4 the diversions due to autoland unavailability, compared to the contemporary triplex system.

A new approach for measuring a system's "failure coverage," based on a failure analysis of a randomly selected set of failure modes, was developed as an outgrowth of the ARCS analysis studies. The approach, shown to be feasible by an analysis/laboratory experiment, is looked upon as a potential cost-effective method for demonstrating a fault-tolerant system's functional success probability.

An assessment of the cost effectiveness of applying ARCS technology was carried out in two parts: an analysis of airline cost-of-ownership for an ARCS maintained in a Cat III operational status and an analysis of the cost effect of providing an integrated system test function. Three primary factors were considered in these analyses: acquisition cost, maintenance cost, and costs associated with schedule interruptions caused by not having an operational Cat II/Cat III system. United Airlines supported these studies by providing a 150-airplane airline model with route, schedule, and cost data associated with operating and maintaining flight control electronic equipment. Study results show a potential annual saving to the airline operator of \$4000 per aircraft when compared to a system representing today's technology.

2.2 STATEMENT OF WORK SUMMARY AND TCV PROGRAM RELATIONSHIP

The criteria development, design synthesis, and design analysis summarized in the previous section were conducted in response to six tasks defined in the ARCS contract statement of work. The objectives of these tasks are summarized below.

The objectives of Task I were to establish design criteria and fault-tolerance management requirements for an advanced reconfigurable computer system. The criteria and requirements were to be applicable to projected future commercial transport avionic systems.

The objective of Task II was to develop a candidate ARCS that overcomes the fault-tolerance performance limitations of the General Electric MCP-703 triply redundant Whole Word Computer System (WWCS) used as a program baseline. The WWCS was developed under Task 4 of the DOT/SST Technology Follow-On Program (contract DOT-FA72WA-2893). In particular, the ARCS development was to result in functional capabilities that achieve the following characteristics of a reconfigurable triple-channel computer system:

- Triple-, dual-, and single-channel redundancy operation
- Automatic computer restart following transient faults (redundancy recovery)
- Automatic signal selection and failure detection with recovery capability
- Automatic system monitoring and automated system test

The objectives of Task III were to assess the WWCS and ARCS configuration operational reliability as a function of fault probabilities, to establish the ARCS operational reliability gains and trends in comparison with the WWCS, and to determine ARCS fault-tolerance performance.

Task IV was to develop system test design criteria, design a system test function, and establish its cost-of-ownership effects when incorporated into the ARCS.

Task V objectives were to determine the fault-tolerant/reliability benefits of ARCS relative to the WWCS and to determine the associated cost benefits of ARCS technology to the airline operator.

Task VI was to assess the feasibility of modifying the WWCS and other suitable commercially available airborne digital computers to conform to the ARCS configuration.

Interactions between the ARCS program tasks are depicted in figure 1. Input to the ARCS development task is the set of projected automatic flight control system design criteria compiled under Task I. The configuration definition resulting from the development task supplies inputs for the cost/benefit assessment, Task V, of the contract schedule. Development of the integrated ARCS test function, Task IV, was conducted in parallel and strongly interacted with Task II. Task III, performance and reliability evaluation, interacted in an iterative way with the design task.

The scope of design application was limited to only flight control functions to obtain a well-defined set of requirements, from one technology, where the need for fault-tolerant processing has been clearly established. Since the most severe system design requirements of a jet transport exist in the flight control area, the general validity of the ARCS program results also applies to other avionic systems applications. On a broad basis, the results of the ARCS program contribute to the Terminal Configured Vehicle (TCV) program and its goals of advancing commercial transport technology.

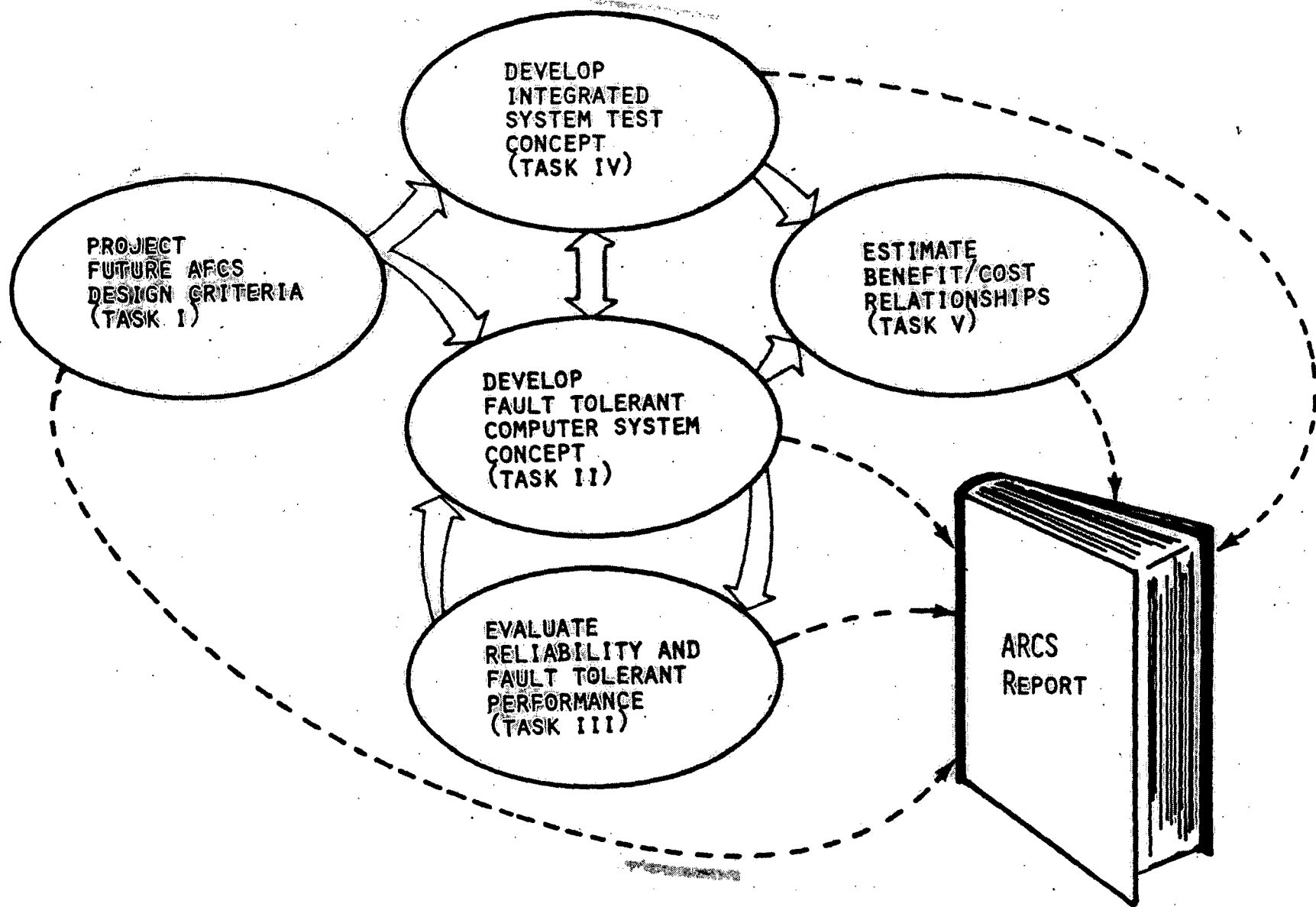


Figure 1.—ARCS Tasks Relationships

Although the ARCS study is a part of the NASA TCV program, the NASA 515 airplane was not considered a suitable application model to govern the development of ARCS design requirements. For this purpose, functional and operational requirements of future, advanced-technology transports were postulated. The TCV program test vehicle, however, could easily serve as a flight test bed for a hardware/software implementation of the ARCS concept resulting from this study.

2.3 REPORT ORGANIZATION

The balance of this report is organized into four major sections, a concluding section, and several appendixes. Section 4 discusses the operational criteria that influence the design of an airborne fault-tolerant computer system, as well as the resulting system design requirements. Section 5 describes the developed ARCS concept from a functional, software, and hardware design point of view. Section 6 discusses the methods and results of the analyses performed with respect to fault-tolerant performance and cost/benefits of the developed concept. Section 7 presents the essence of a fault-tolerant computer system specification and discusses the results of evaluating the adaptability of existing computer systems to the ARCS specification. The concluding section, section 8, summarizes the study results and provides concluding comments on the overall ARCS program.

To maintain visibility of the major themes presented in the main report, extensively detailed material is presented as appendixes. Application control laws, software design details, hardware design analysis, simulation results, and reliability analysis details are examples of material contained in an appendix.

2.4 ACKNOWLEDGMENT

The Boeing Commercial Airplane Company wishes to acknowledge the General Electric Company, Aerospace Controls and Electrical Systems Department, and United Airlines, Maintenance Operations, for their support during the ARCS program. Messrs. L. E. Fairbanks and R. P. Kurlak of GE deserve recognition for their contributions to the system design/analysis tasks, and Mr. H. Takeuchi of United Airlines is recognized for his contribution of airline operational parameters to the system test design and cost/benefit assessment tasks.

3.0 SYMBOLS AND ABBREVIATIONS

AC	Advisory Circular
A/D	analog to digital
AEEC	Airline Electronic Engineering Committee
AFCS	automatic flight control system
ARCS	Airborne Advanced Reconfigurable Computer System
ATA	Air Transport Association
ATC	air traffic control
ATE	automatic test equipment
ATT	Advanced Technology Transport
BCAR	British Civil Airworthiness Requirements
BIT	built-in test
BITE	Built-in test equipment
CARSRA	Computer Aided Redundant System Reliability Analysis
Cat II	Category II
Cat III	Category III
CCV	controls-configured vehicle
CDP	control and display panel
CIU	computer interface unit
CLR	clear (instruction)
CPU	central processor unit
CRT	cathode-ray tube
CTOL	conventional takeoff and land
CWS	control wheel steering

D/A	digital to analog
DADS	digital air data system
D/D	digital to digital
DELS	direct electrical linkage system
DG	directional gyro
DMA	direct memory access
DME	distance measuring equipment
DOIO	directly operable input/output
DOT	Department of Transportation
DRO	destructive readout
EIA	Electronic Industries Association
EMI	electromagnetic interference
FAA	Federal Aviation Administration
fail-op	fail-operational
FAR	Federal Aviation Regulations
FBW	fly-by-wire
FF	flip-flop
FIFO	first in first out
FMEA	failure mode effect analysis
GE	General Electric Company
GSE	ground support equipment
HLH	heavy lift helicopter
ILS	instrument landing system
I/O	input/output

INS	inertial navigation system
ISADS	integrated strapdown air data system
kops	thousand (kilo) operations per second
K	thousand words of storage
LED	light-emitting diode
LIFO	last in, first out
LRU	line replaceable unit
LSB	least significant bit
LSC	local sync command
LSI	large-scale integration
LVDT	linear variable differential transformer
MCP	modular control processor
MEL	minimum equipment list
MLS	microwave landing system
MPX	multiplex
MSB	most significant bit
MTBF	mean time between failure
PFCS	primary flight control system
PROM	programmable read-only memory
R/A	radio altimeter
RAM	random access memory
RCSS	Reconfigurable Computer System Simulation
RESRA	Redundant System Reliability Analysis
R-NAV	area navigation

ROM	read-only memory
SAS	stability augmentation system
S/H	sample and hold
SLI	synchronous logic interface
SMAR	serial memory address register
SOV	shutoff valve
SPBP	split-phase bipolar
SSFD	signal selection/failure detection
SST	supersonic transport
STOL	short take-off and landing
STP	system test panel
STRU	servo transmit/receive unit
TCV	Terminal Configured Vehicle
TMR	triple-modular redundancy
TTL	transistor-transistor logic
UA	United Airlines
UART	universal asynchronous receiver/transmitter
VG	vertical gyro
VOR	very high frequency omnirange
WWCS	Whole Word Computer System
XMIT	transmit (instruction)
2-D, 3-D, 4-D	two-, three-, four-dimensional

4.0 DESIGN CRITERIA

A design requirement or design goal is normally expressed as, or associated with, a criterion, i.e., a standard, rule, or test for measuring the excellence, fitness, or correctness of the particular characteristic specified. Airplane profitability and flight safety are the two criteria that ultimately dictate the design of an airborne processor system used for flight control in a commercial transport.

Design requirements and design goals for a subsystem must necessarily reflect requirements and goals formulated on the higher system level. Thus, identifying design requirements for the computer system starts with defining ARCS-related requirements and goals formulated on the airplane operation level. The criteria defined on the operational level are subsequently interpreted and formulated into criteria applicable to the design of the computer system, as illustrated by figure 2.

Past experience of analog and digital flight-safety-critical computer systems is integrated into the interpretation process. Other influences stem from the adopted goals and objectives of the ARCS program as expressed by the contract statement of work. The most far reaching of these goals is to achieve high fault tolerance by providing the capability for a triplex system to degrade in redundancy into an operational simplex system configuration.

The computer system is a subsystem of the total airborne vehicle; it interfaces or interacts with other subsystems in the airplane. Design optimization of the reconfigurable computer system cannot be accomplished out of context, i.e., without considering the total vehicle requirements and the functional and operational environment of the computer system; since the interfacing or interacting subsystems contribute to the final outcome of system reliability and fault-tolerance performance trades, they must be adequately represented in the analyses. A significant portion of establishing the ARCS design criteria therefore involves identifying the functional context and the interface environment into which an operational ARCS will be integrated.

4.1 OPERATIONAL CRITERIA

A continuing evolution in flight procedures and airframe, sensor, and electronic circuitry technologies will take place during the timespan in which an ARCS would become applied to an airplane. Advancements in airframe and sensor technologies will influence the functional complement of an ARCS: the further into the future the requirements are projected the more advanced functions the fault-tolerant computer system will be required to perform. From a commercial transport design point of view, considering functional reliability and mission flight safety, a controls-configured vehicle (CCV) type of fly-by-wire (FBW) is the most advanced type of function to be anticipated for the foreseeable future. The autoland function, although implemented in aircraft today, represents another significant design point with respect to the design of a highly reliable, fault-tolerant computer system.

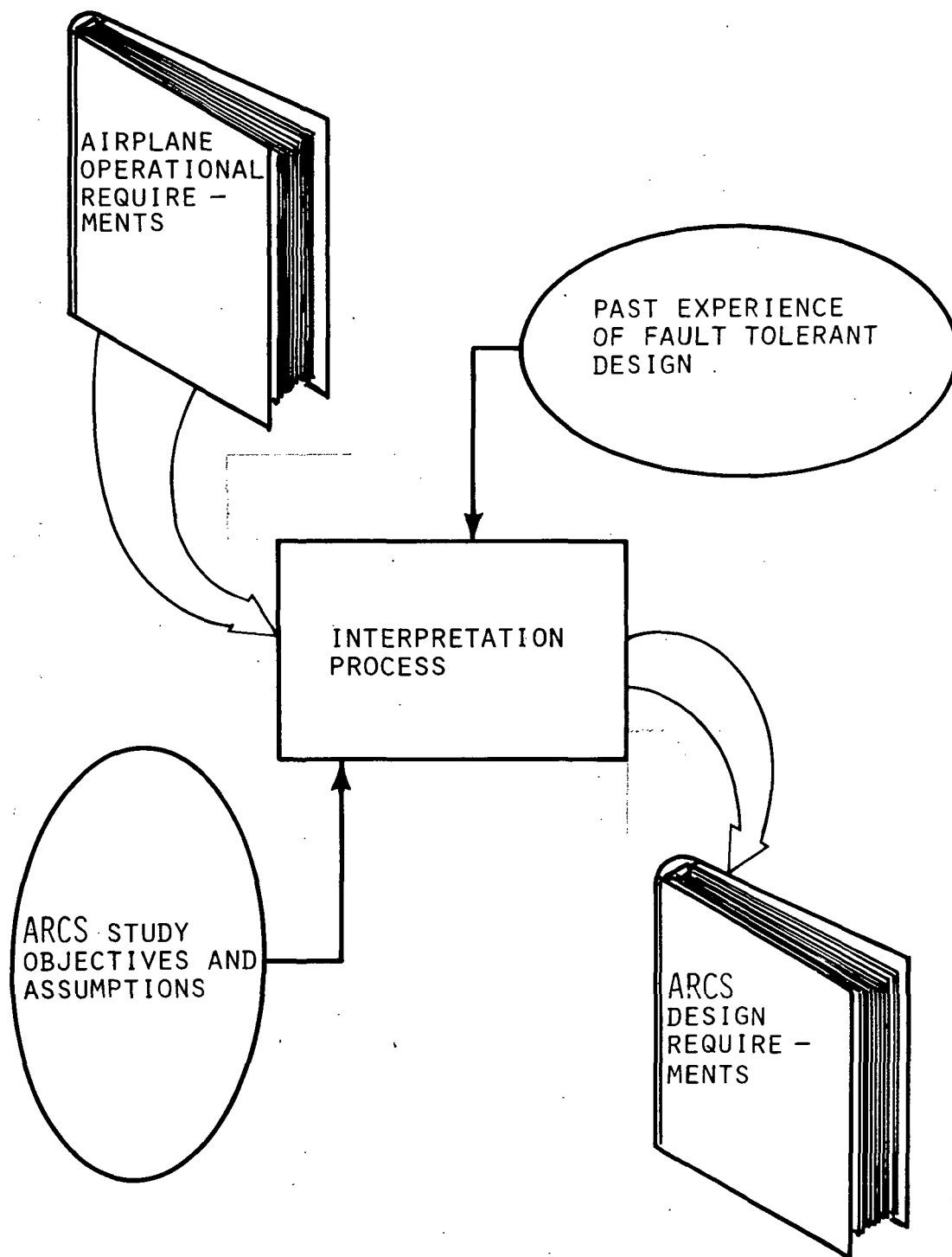


Figure 2.—ARCS System Requirements

The advancement of electronic circuitry technology, including the emergence of airborne qualified microprocessors, will impact the physical aspects (piece parts) of the ARCS hardware much more than it will the functional architectural aspects. The system functional architecture will be dictated largely by flight control design considerations.

For the purpose of defining the ARCS design criteria, it was postulated that the CCV/FBW and autoland functions would be by far the most critical functions to be implemented in the ARCS. Therefore, specifying reliability and fault-tolerant performance levels for these two functions would automatically ensure adequate reliability and performance levels for other ARCS functions.

To adequately treat the advancements in sensor and airframe technologies, it was further postulated that three application models be used for the ARCS development: a near-term model, using existing sensors, to provide the basis for comparison with the baseline WWCS technology; an intermediate-term model based on a step advancement in sensors, postulating a digital, integrated air-data strapdown sensor system; and a far-term model that anticipates the full implementation of CCV/FBW airframe technology.

The airplane operational requirements pertinent to the design of an ARCS are discussed in the following two sections. They break down into two groups, as illustrated by figure 3: criteria derived as a result of economic considerations, i.e., those addressing operational profitability (sec. 4.1.1), and criteria established to ensure adequate flight safety (sec. 4.1.2).

The ARCS design requirements resulting from interpretation of the operational requirements are presented in section 6. These design requirements are divided into three groups addressing the functional design, software design, and hardware design.

4.1.1 ECONOMIC CONSIDERATIONS

Except for equipment required by regulation for flight safety, no new avionic systems are installed in airplanes without expectations of improved overall airplane profitability. There are three areas in which an ARCS has the potential to improve aircraft cost efficiency:

- The *functional scope* of the system, which influences the initial system cost as well as the potential operational benefits of the system
- The *functional readiness* of the system, or functional availability, which determines the degree of achieving the potential operational benefits
- The *system maintainability*, which influences the burden of maintaining the system at the functional status required to achieve the operational benefits

How do these three characteristics translate into design requirements for the computer system; what are the relationships between operational and design criteria?

The *functional scope* of the flight control system determines the computational speed and memory capacity requirements, as well as the sensor, servo, and mode control/display interface requirements, that the fault-tolerant computer system must satisfy.

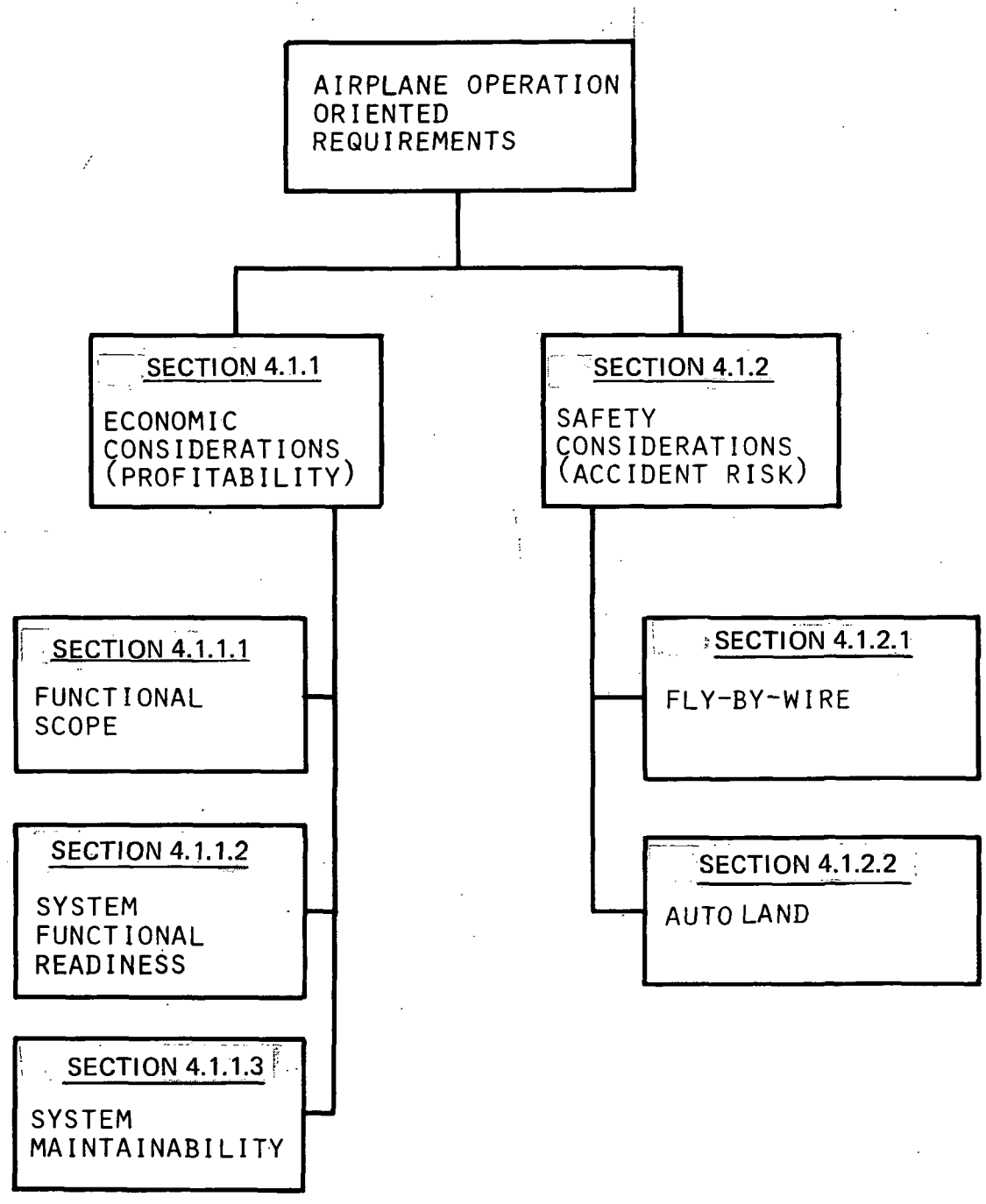


Figure 3.—Airplane Operational Requirements Breakdown

The *functional readiness* or availability translates directly into the computer system's capability to survive failures and to handle multiple component failures in the interfacing sensors and servos; it will drive the designer to seek a solution with multifailure tolerance.

The operational *system maintainability* requirements will result in a set of design requirements aimed at the implementation of computer built-in test (BIT) functions for the computer itself and for its interfacing peripherals, including the man/machine interface for the test functions.

The following sections discuss the above three areas and identify operational requirements for the definition of ARCS design requirements.

4.1.1.1 System Functional Scope

From an airline operational point of view, automatic flight control functions fall into one of the following three categories:

- *Dispatch-Critical Functions*—functions without which an airplane cannot legally be dispatched on a revenue flight
- *Flight-Mandated Functions*—functions without which certain operations cannot be conducted and without which an airline may elect not to dispatch an airplane under certain conditions
- *Pilot Workload Relief Functions*—functions that do not impact an airplane's dispatch status nor significantly affect the flight plan, yet have a definite convenience value to the pilot

From the flight control system designer's point of view, functions in these categories belong in three different categories (see fig. 4):

- *Flight-Crucial Functions*—functions without which an immediate unconditional flight safety hazard exists
- *Flight-Critical Functions*—functions without which a potential short-term flight safety hazard exists conditioned upon factors such as environment or flight condition for which the pilot has the option to circumvent by taking an operational penalty (flight diversion)
- *Noncritical Functions*—functions without which no short-term impact on flight safety exists

The ARCS design requirements identified herein project an increasing reliance upon automation and active controls technology to improve the productivity of transport airplanes. The increasing utilization of more flight-crucial functions with time, reflected by the three ARCS application models defined below, is based on an expectation of continued evolution of sensor and electronic hardware technologies, resulting in improved reliability and cost.

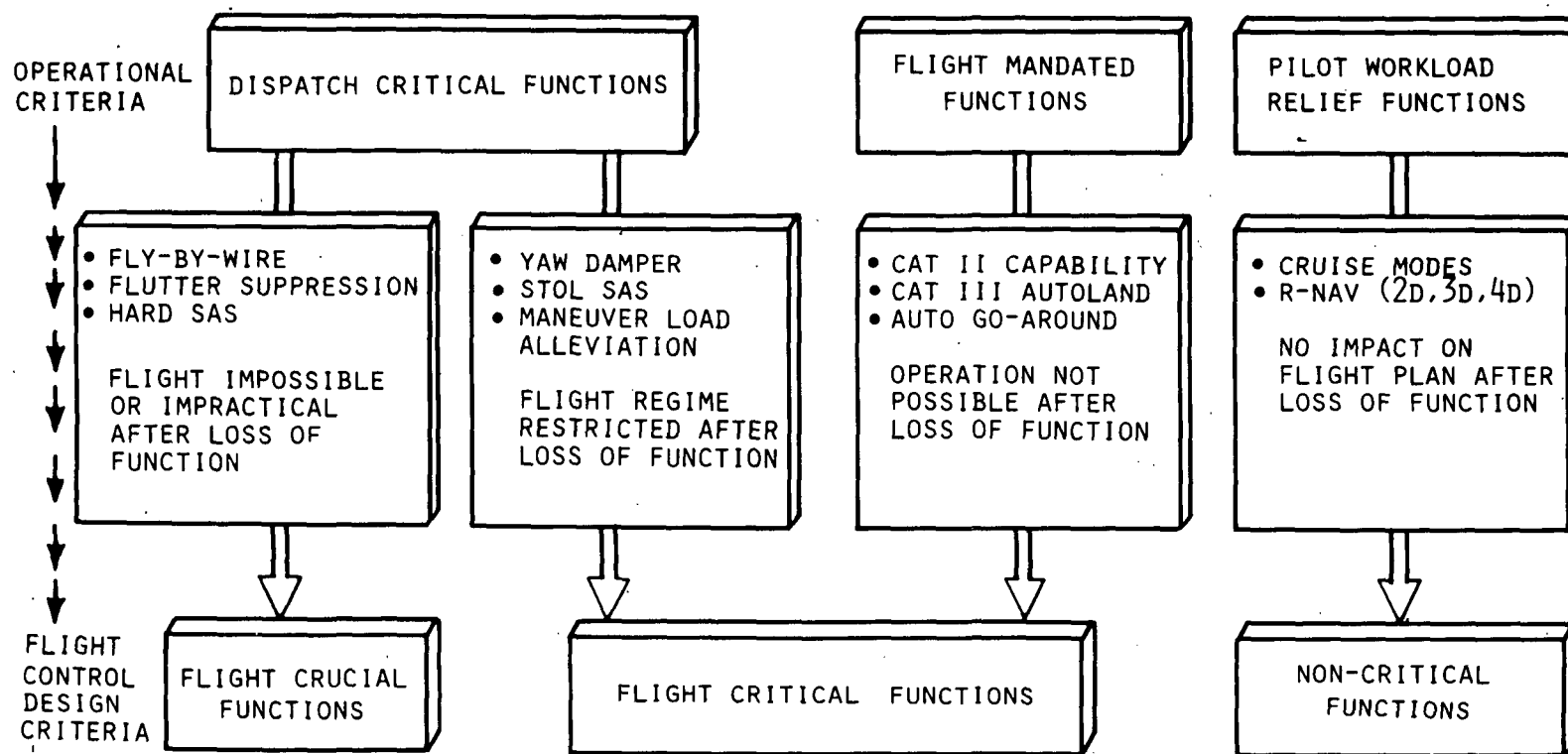


Figure 4.—Operational/Design Criteria Relationship

The general system interface architecture, shown in figure 5, shall be capable of handling simplex, duplex, triplex, or quadruplex redundancy, although at the present time the required redundancy level cannot be established for all anticipated applications with a high degree of confidence. The basic ARCS architecture was developed for a triplex system configuration.

The projection of the functional scope of each of the ARCS application models is based on assumptions relative to the technological development of sensor, airframe, and electronic circuitry technologies at three time periods designated as near term, intermediate term, and far term, as illustrated by figure 6. The near-term application is characterized by today's existing sensors and commercial transport airframe technology. The far-term model represents airframe technology beginning to be implemented in production fighters today, which implies historically that it is 10 to 20 years in the future for commercial transport application. The intermediate term is a step in between, which takes advantage of new sensors currently under development that simplify system interfaces and allow a more efficient system organization through digital data communication.

The assumptions made for sensor and airframe technology advancement, and the functional capabilities deemed to be desirable and justified for each application time period, are discussed below. Typical control law configurations for the flight-critical functions and interface requirements are given in appendix A.

Near-Term Application Model.—The near-term commercial transport being designed today will be assumed to have a basically stable airframe, the only probable dispatch-required automatic flight control function will be a yaw damper. This conservative assumption reflects the Boeing design philosophy for an airplane that would be introduced in 1980. The primary flight control system will be mechanical or direct electrical links not requiring sensor feedback processing.

The sensors available for the near-term application will be those sensors available and proven today. The traditional organization of automatic flight control separated from navigation/guidance functions will prevail, and no integration of mode select panels for those functions will be assumed. An area-navigation (R-NAV) computer will provide steering command via serial digital bus to the flight control computers, and the air data computer will be digital.

Attitude and heading information from vertical/directional gyros (VG/DG) will be standard, with the inertial navigation system (INS) being an available option for long-range aircraft. Roll and pitch rate gyros will not be included; rates will be derived from attitude information.

The flight control mode-select panel will communicate with the computer via discretes for flight-critical functions and via serial digital links for noncritical functions. The near-term application model will utilize miniprocessors.

The functional complement of the near-term model will consist of the following capabilities:

Flight-critical functions:*

Category IIIb autoland
Low-speed yaw SAS

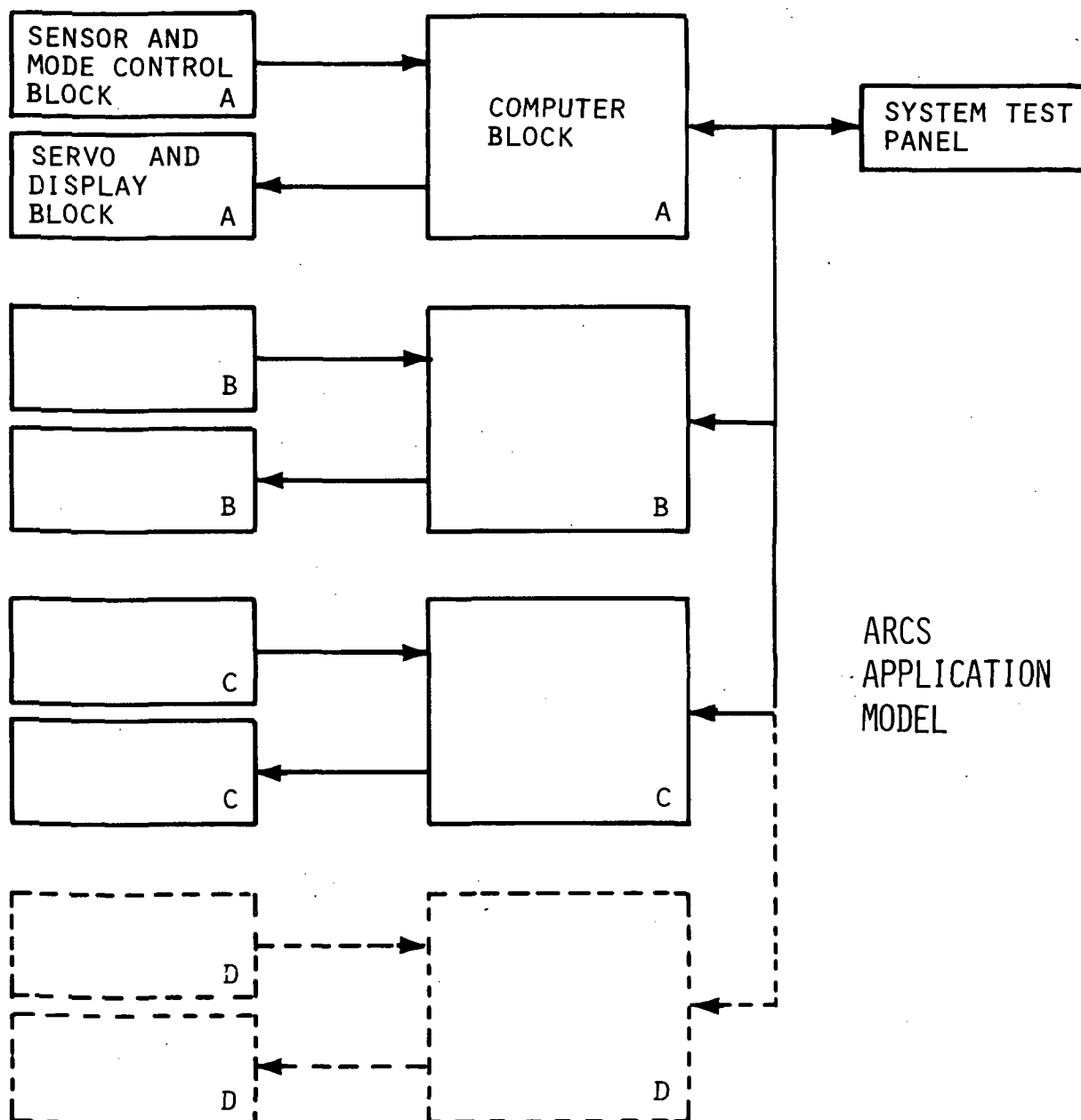


Figure 5.—General System Interface Architecture

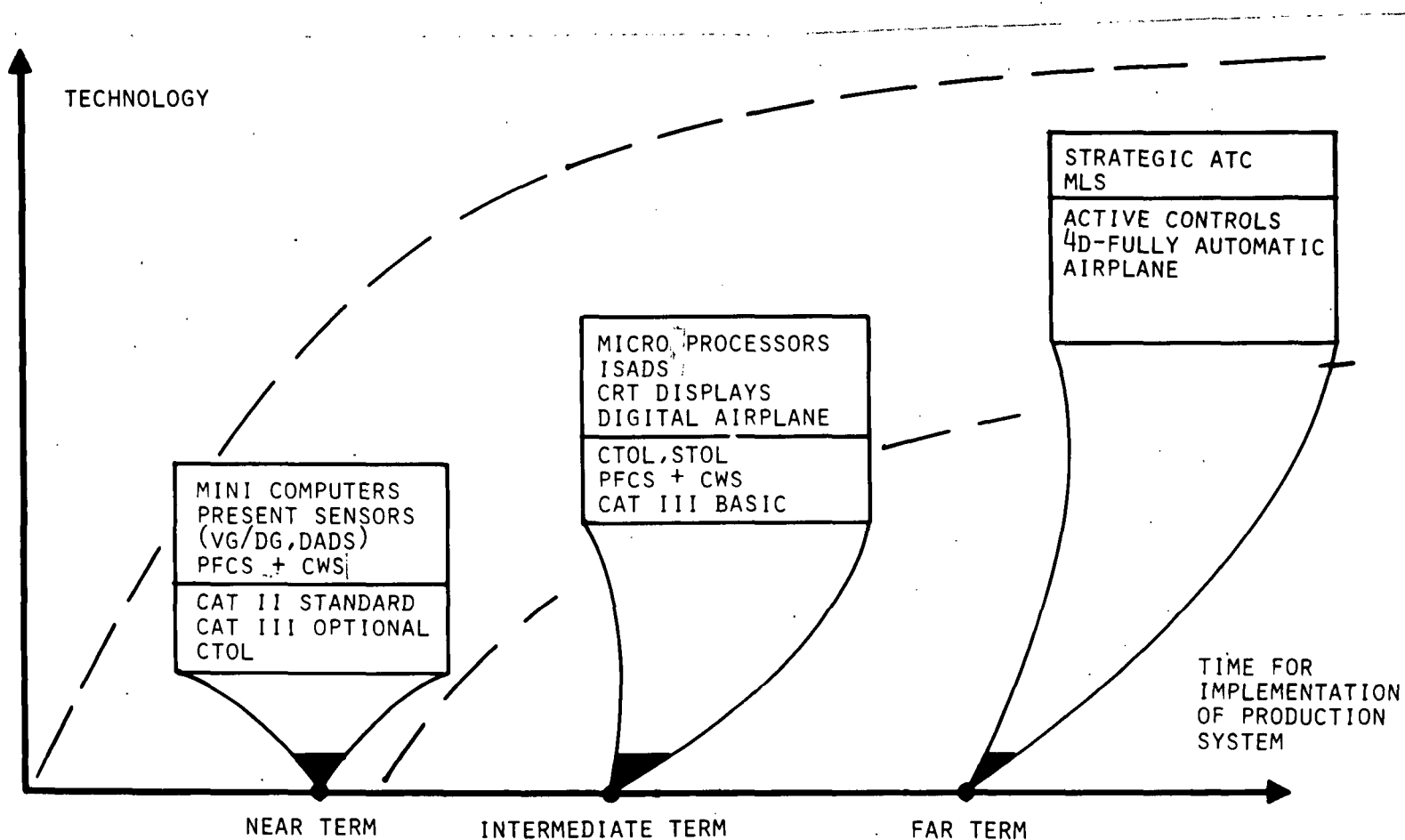


Figure 6.—ARCS Application Models

Noncritical functions:

Control wheel steering (CWS)
Altitude hold/select
Airspeed hold/select
Mach hold
Vertical speed hold/select
Heading hold/select
2-D/3-D R-NAV coupler

*High-speed yaw SAS will be a separate analog system. This will be the only dispatch-required AFCS function.

Intermediate-Term Application Model.—The intermediate-term airplane will be assumed to have taken half the step toward fly-by-wire; limited-authority series servo control augmentation will be used in all three control axes, but an unaugmented reversionary mode (mechanical or direct electrical link) will be available for operation within a restricted flight envelope. This airplane could be a STOL transport or a CTOL transport with relaxed basic airframe stability requirements; an operating automatic flight control system would therefore be a dispatch requirement.

The major improvement in sensor technology for this application model will be the availability of an integrated strapdown air data system (ISADS), providing all attitude, rate, acceleration, and air-data-derived signals in a digital format. In addition, a digital low-range radio altimeter with improved reliability is assumed to be available. Except for airplane-mounted force and position sensors and instrument landing system (ILS) deviations, all sensor signals will be in digital format. A digital display system using CRT displays will be assumed.

A completely integrated approach to the cockpit design and the system-to-pilot interface has been assumed for the intermediate-term model. All noncritical flight control functions are assumed to be performed by the navigation-guidance computers, as indicated by figure 7.

The functional separation shown in figure 7 represents the anticipated organization of a future integrated navigation/guidance/flight control system for a commercial transport. The functional grouping provides for flight-critical functions in a highly reliable, fault-tolerant computer system with simple, redundant, electrically separated man/machine interfaces, and noncritical functions in lower redundancy with nonredundant, more complex man/machine interfaces. The configuration described by figure 7 has evolved as a result of the experience in fault-tolerant flight control systems and integrated avionic systems research, development, and testing conducted by Boeing during the past several years.

A microprocessor (LSI) implementation of the ARCS processor will be assumed possible for the intermediate-term model. Reliability and failure effect considerations must determine the optimal approach to hardware implementation.

The functional complement of the intermediate-term application model will be as follows:

Flight-critical functions:

STOL SAS
Full-time command/stability augmentation
Cat IIIB autoland

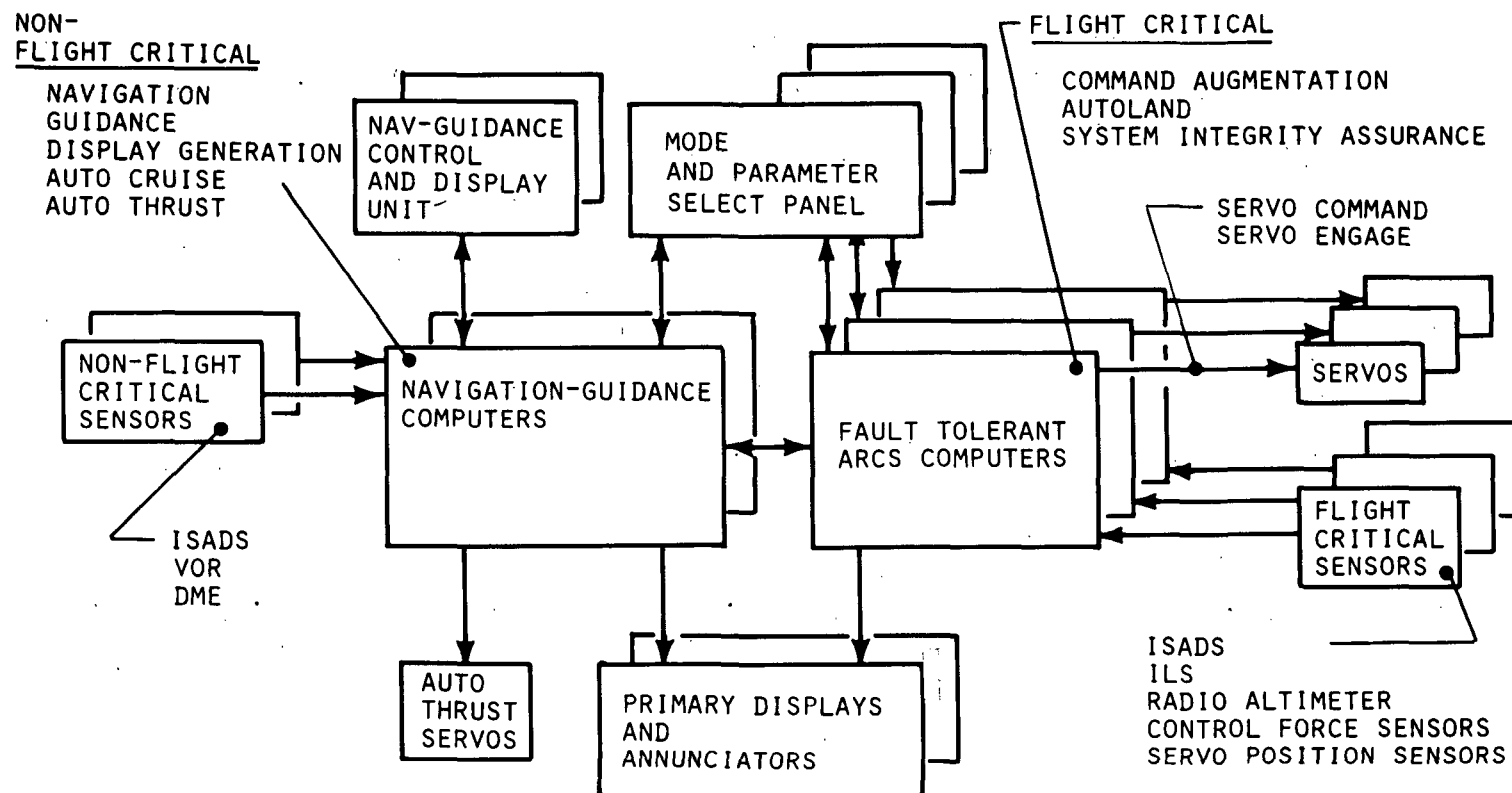


Figure 7.—Integrated Navigation/Guidance/Flight Control System

Noncritical functions:*

Track angle select/hold
Flightpath angle select/hold
Autothrust modes
2-D/3-D/4-D command generation

*Performed by nav-guidance computer

Far-Term Application Model.—For the far-term application model, an Advanced Technology Transport (ATT) concept was selected. This is a controls-configured vehicle/fly-by-wire (CCV/FBW) airplane with one pitch control surface, two pairs of roll control surfaces, and two pairs of structural mode/flutter control surfaces.

The only additional sensor assumed available for this application model is the microwave landing system (MLS) receiver. The structural mode and flutter suppression requires accelerometers and rate sensors distributed in the structure of the vehicle. Except for these structural mode sensor signals, the control force sensor signals, and the servo position and differential pressure sensor signals, all signals will be either digital or discrete.

The total system organization for this application model will be similar to the configuration shown in figure 7, with the flight-crucial functions included in the fault-tolerant computer. It is expected that a basic redundancy level of quadruplex will be required for the CCV/FBW application. Full use of microprocessor technology is anticipated in order to contain failure effects and increase system functional survivability.

The functional complement of the far-term application model was assumed to be as follows:

Flight-crucial functions:

Flutter suppression
Structural mode suppression
Fly-by-wire control
Full-time stability augmentation

Flight-critical functions:

Category III MLS autoland

Noncritical functions:

Same as for intermediate-term model,
plus air-ground data link for ATC
communication; performed by nav-
guidance computer

Aircraft Transient Fault Environment.—Transient phenomena emanating from the airplane environment can temporarily disable one or more redundant elements in the flight control system and must be considered in the design of the ARCS, whether for the new-, intermediate-, or far-term applications. Four potential sources of such transients exist in a commercial airplane: electrical power bus voltage variations, electromagnetic pulses generated by lightning strokes, transient sensor signal disagreements, and hydraulic power system pressure transients.

Power transients on aircraft electrical power buses occur for two reasons. Normal transients are caused by normal switching of loads in the course of operating the aircraft. Abnormal transients are caused by failures of loads on a bus. Current power quality specifications for new airplanes require that normal transients have a duration of less than 0.1 second and that automatic fault clearing keeps abnormal transients to a duration of less than 10 seconds. Up to 20 normal power transients per hour can be expected during flight operations.

Lightning strokes on Boeing aircraft have occurred on the average of once per 2500 hours. The duration of the high-energy transient in the discharge is assumed to be no greater than 50 μ s.

Transient sensor signal disagreements can occur for two reasons. Static gain differences in combination with large aircraft maneuvers will cause sensor signal differences, even though the sensor outputs are within specified tolerances. Tolerances in dynamic response characteristics between like sensors, in combination with installation variances and turbulence, will likewise result in disparities between nominally like signals.

Transient pressure variations in the hydraulic power systems are caused by large load applications. Transient pressure and servo position differences between redundant flight control elements are conceivable since individual hydraulic systems carry different complements of loads.

The ARCS design shall be structured to remain operational for, and recover from, faults induced by the transient phenomena described above.

4.1.1.2 System Functional Readiness

Component failures will occur in the flight control system with an average frequency determined by the component failure rates. The probability that a given function will remain operational, or be operational when called for, decreases with the exposure time since the last functional verification.

For the ARCS program, the following definition for functional readiness was made:

Functional readiness is the probability of retaining a function at a certain time assuming the system is initially fully operational.

The calculated functional readiness for an autoland function is illustrated by figure 8. The lower curve represents a triplex fail-operational system composed of present-day sensor, servo, and airplane system technology components and a triple-modular-redundant (TMR) computer system with a failure rate equal to that of the baseline WWCS. The upper curve illustrates the ultimate improvement in functional readiness that can be achieved if a system with identical component failure rates can be designed to survive two like failures, i.e., can degrade to simplex operation for any two like failures.

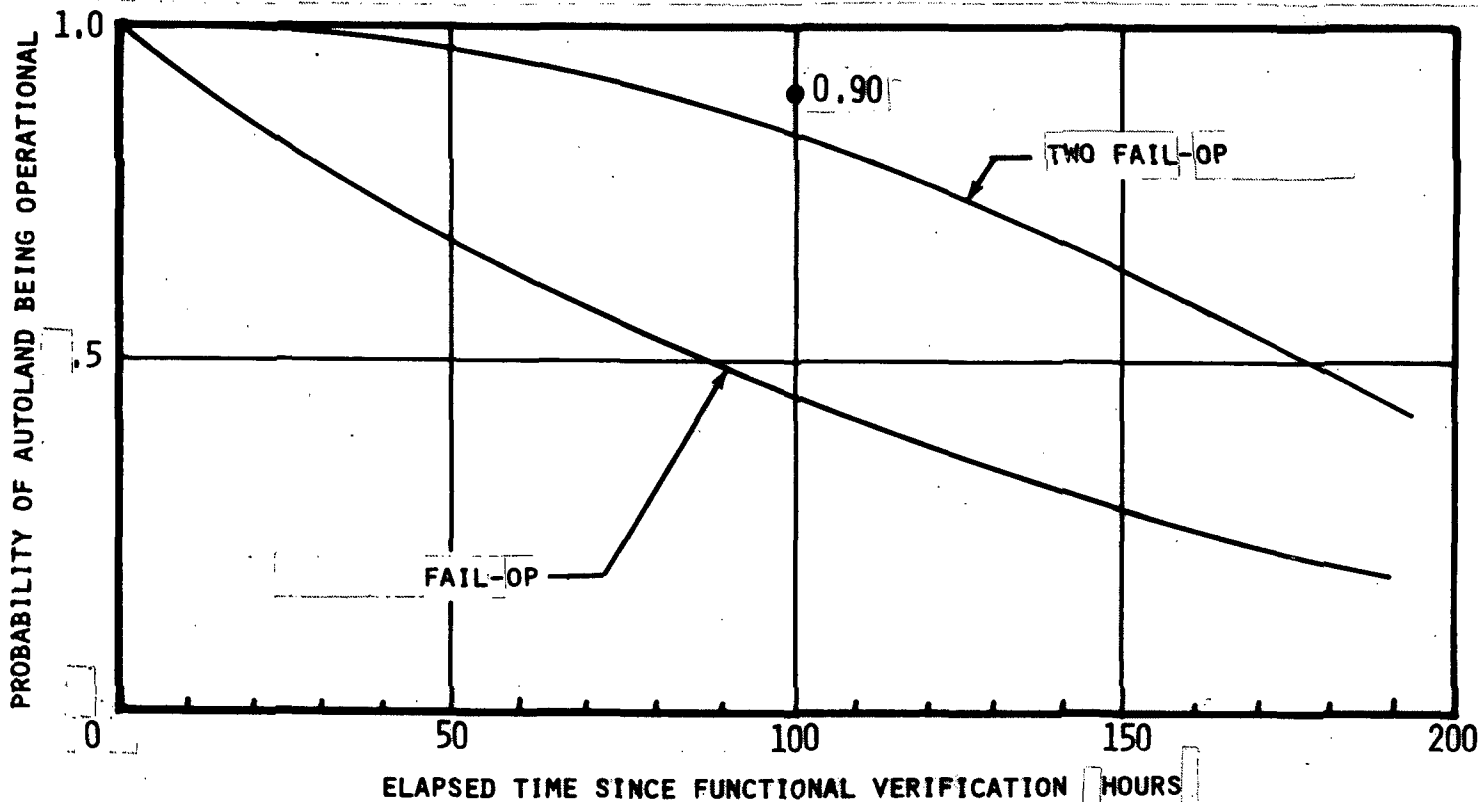


Figure 8.—Autoland Functional Readiness for Triplex System

For the reliability calculations illustrated by figure 8, constant component failure rates were assumed. This implies that elapsed time since last verification of fault-free performance can be substituted for elapsed time since maintenance. Fault-free performance could be verified by preflight test (typical for FBW) or by the fault-free use of the function on a previous flight (autoland).

In addition to functional readiness, the concept of function availability is useful for the specification of ARCS criteria. It was defined as follows:

Function availability is the probability that a function is available for use at any point in time, given a certain maintenance schedule.

Availability, with this definition, is the average functional readiness over the operating time-span considered. For the ARCS, the operational timespan of interest is a series of airplane flight missions from dispatch to gate arrival at the airplane's destination. This is compatible with the ARINC definition of availability, where the times for active repair of a system, logistics, and administration are ignored in the context of specifying the reliability of a given functional capability.

To achieve acceptable functional readiness and availability, certain operational and economic criteria must be established. Due to the different criticality of the two major ARCS functions, functional readiness criteria take different forms for the FBW and the autoland functions, as described below.

CCV/FBW Functional Readiness.—The very nature of the flight-crucial CCV/FBS function makes it dispatch required; its functional availability at dispatch is therefore set at unity by edict. However, to ensure an economical design, specific maintenance requirements must be set forth to limit the cost of maintaining the system in its dispatch-required condition.

There is a direct relationship between maintenance effort and frequency of line replaceable unit (LRU) failures. Since every failure in the dispatch-critical function has to be repaired before the next flight, a readiness curve for a FBW function, corresponding to that shown in figure 8 for the autoland function, must start at time zero for every flight where the functional readiness is unity. The only criterion availability to assess the acceptability of the FBW functional readiness is the maintenance effort required to keep the FBW functional readiness at unity for every flight dispatch.

The first functional readiness requirement for the CCV/FBW function thus addresses the cost of maintaining the system. Lacking specific requirements in this area, the following position was taken for the ARCS:

The effort required to maintain the FBW function in dispatch-required status shall not exceed the present level of maintenance for dispatch-required flight control functions.

The level of maintenance required for primary flight controls (ATA Chapter 27) on airplanes operational today is approximately 8 man-hours (line and shop maintenance) and \$50 of material cost per 100 hours of flight time.

The second FBW functional readiness criterion addresses the situation when a first failure state in the FBW system has been incurred. This condition, which can occur anytime after the airplane is dispatched from the departure gate and which has a reasonable probability to happen in a complex function like the CCV/FBW, must not impair the operational utility of the airplane. The following requirement was therefore postulated:

No first (in-flight) failure of components involved in the FBW function shall require that the flight plan be modified for any flight duration of less than 10 hours.

Ten hours was selected to be representative of the longest flight time presently anticipated for commercial transports.

In the event several components fail within the CCV/FBW function, a failure condition could arise that requires the airplane to land within a short time so that safety-of-flight requirements based on survival probability of the partly failed system are not violated. Any such diversion constitutes a disruption of the airline operation; therefore, the third FBW functional readiness criterion addresses the diversion probability. The position was taken that diversion probability due to FBW failure must not be higher than the current diversion probability due to primary flight controls (ATA Chapter 27):

Any failure state of the FBW function requiring diversion from the flight plan shall be improbable.

Improbable shall be interpreted consistent with its use in the FAR's. Although not defined quantitatively in any regulatory document, a probability lower than 1×10^{-5} has been used by industry to signify an improbable event. The average diversion rate per departure due to flight control system failure in the United Airlines fleet presently is approximately 3×10^{-5} .

Low-Visibility Automatic Landing Functional Readiness.—The Category III autoland function is not anticipated to become dispatch required in the foreseeable future. It is conceivable, however, that given demonstrated operational benefits, an airline may declare the function flight mandated, i.e., an airplane will not (normally) be dispatched without repair if the autoland function is known to have incurred a failure state that would disqualify it from being used on the next landing.

The lower curve of figure 8 also represents the predicted functional readiness of an existing fail-operational triplex analog Category IIIa autoland system currently in airline service. The actual functional readiness experienced in service with this system closely follows this curve. This functional readiness history is considered unacceptable on operational and economic grounds by at least two major international flag carriers. For long "thin" international routes in particular, aircraft flight time on the order of 100 hours before return to the airline's primary maintenance base is not uncommon.

After reviewing these international airline operations and considering our experience with current systems, it was postulated that a low-visibility autoland function, in order to be operationally and economically justifiable, must have a function availability goal as follows:

The average probability of having an operational capability required to commit to a Category III landing shall be no less than 0.9 for 100 hours between system maintenance.

The number 0.9 was chosen from airline comments on desired Cat III availability; no recognized airline or FAA standard relates to Cat III availability.

4.1.1.3 System Maintainability

Fault-tolerant redundant automatic flight control systems have been introduced into airline operation to prevent hardware failures from creating hazardous situations during a critical operation such as autoland. Redundancy is also being used in aircraft systems to enhance basic system functional availability. While these systems have been developed to achieve desired safety and functional reliability goals, many complex problems have surfaced concerning testing and maintaining system operational integrity. These problems have offset to an uncertain extent the economic benefit of introducing the higher functional capability.

Fault tolerance requires that faults occurring within the system be detected, localized, and isolated. The ability to localize faults therefore becomes an inherent requirement for a system designed to tolerate multiple like faults. As a consequence, the system design must include mechanisms to test for, and register the existence of, fault conditions as they occur in operation. In addition, if the computer system is to be tested for the existence of latent failures, provisions must be made for conducting such a test in an off-line or non-real-time mode.

The maintenance problem for automatic flight control systems (AFCS) is more complex than for most other avionic systems. To function properly, the AFCS requires other interfacing aircraft systems such as sensors, servoactuators, electrical power generation/distribution systems, and hydraulic power systems. A problem with any one of these interfacing systems can potentially result in degradation in AFCS performance capability, thus requiring that maintenance action be taken. Experience with contemporary equipment shows that to maintain such complex, interdependent systems using manual testing and troubleshooting techniques is a formidable, time-consuming airline maintenance task, and maintenance effectiveness has fallen far short of desired goals.

The magnitude and criticality of the AFCS maintenance problem causes increasing concern throughout the aircraft industry. Evidence of this concern is given by the recently formed Airline Electronic Engineering Committee (AEEC) Subcommittee on Built-In Test Equipment (BITE), whose major undertaking thus far has been to address the area of built-in test considerations for future digital automatic flight control systems. Furthermore, with the current economic pressure on the airline operators, it is anticipated that the need to improve maintenance efficiency will persist.

The identified requirements—on-line fault detection for fault tolerance and improved maintenance effectiveness—led to recognizing the need for an improved, integrated system-level self-test/diagnostic capability. An automated system test function shall satisfy these two requirements of the ARCS.

System test, as applied to the ARCS, can be viewed as an automated built-in test and diagnostic function designed to provide fault detection and LRU-level failure isolation capability for the AFCS including sensors, computers, servoactuators, and interface elements. The ability of system test to rapidly and comprehensively test the various facets of the ARCS provides a high confidence in system operational integrity, which is an integral element of fault tolerance. Furthermore, by automating the maintenance decision-making process, system test will not only reduce system maintenance time requirements but also improve overall maintenance effectiveness, thereby reducing system ownership cost.

The following subsections identify the criteria and requirements for the maintenance-related system test function. These requirements, developed by United Airlines, were discussed and generally supported by members of the AEEC BITE subcommittee. The criteria and requirements discussion has been subdivided into five categories: maintenance test conditions and efficiency, operational constraints, test procedures, fault detection and recording, and system test/crew interface.

Maintenance Test Conditions and Efficiency.—Cost effectiveness is the most important parameter to consider in the development of maintenance enhancement methods. A primary factor in assessing system cost effectiveness is the number of maintenance man-hours required per flight-hour, where maintenance man-hours include both line and shop maintenance. This consideration is the basis for many of the ARCS system test design requirements. For example, any need for periodic or time-controlled maintenance places an unnecessary cost burden on the maintenance operation, no matter how simple the tests may be. A 5-minute test required on each one of United Airline's 600 000 annual flights would add up to 50 000 maintenance man-hours. Because of this, airlines in general have adapted, or are converting to, a maintenance program based on the "condition monitoring" philosophy. Under this program, maintenance action is initiated only as a result of a failed preflight test, if applicable, or an in-flight failure reported by the flightcrew. A necessary consequence of this philosophy is that unless a preflight test is required or unless a squawk was generated on the previous flights, the system is assumed to be operational and available for service. In keeping with this philosophy, ARCS maintenance will be conducted on strictly a "condition monitoring" basis.

An assessment of maintenance practices and experience on contemporary flight control systems highlights one major expense item which, perhaps more than any other, is indicative of the inadequacies of present maintenance aids and procedures: the "unverified" equipment removals. A removal (normally incurred in response to a reported flight problem) becomes an unverified one when either the same squawk occurs on subsequent flights or the removed unit is found to be fault-free in the shop. Unverified removals are typically a result of so-called "shotgun" maintenance techniques, wherein equipment is removed and replaced in an effort to clear a flight squawk without adequate testing and troubleshooting. Typical unverified removal rates for contemporary flight control system LRU's range from 45% to 70%. These excessive unconfirmed removal rates are costly not only because of the unnecessary line and shop maintenance action but because of the expense associated with the additional pipeline spares.

Test effectiveness was defined as the ratio of faults detected to faults incurred. The effectiveness goal for the ARCS system test function in detecting and localizing failures within the computer subsystem has been placed at 99% for a redundant configuration and no less than 95% in a simplex configuration. The effectiveness goal of the system test function in detecting and localizing failures within the redundant sensor and servo systems has been placed at 90%. These quantitative goals are based on experience with inertial navigation systems and certain digital flight control systems used for military applications. Such goals are in keeping with current airline maintenance operations as expressed at a recent BITE subcommittee meeting and illustrated in figure 9, which shows that approximately 50% of the AFCS squawks

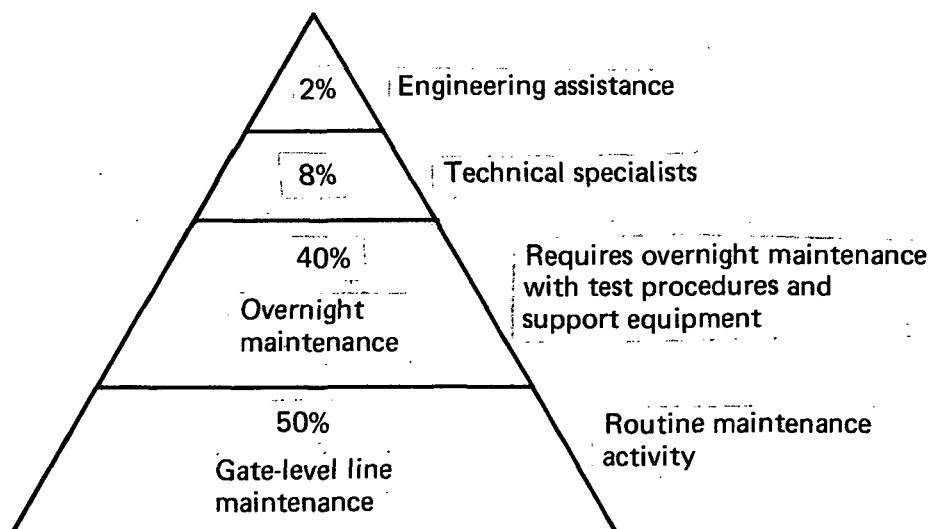


Figure 9.—Maintenance Effectivity

are satisfactorily repaired by routine line maintenance, 40% are resolved during an overnight repair cycle, 8% require the intervention of specially trained technical specialists, and 2% require engineering action. The system test function is expected to be effective between the 90% level achievable by line maintenance (turnaround or overnight work) and the 98% level currently achieved by line maintenance plus technical specialist.

The interest of airline operators is not in the percentage of the circuitry covered by system test but rather in the percentage of equipment removals that are confirmed. This consideration imposes two requirements on the design: (1) consideration must be given to failure rates when configuring a system test and (2) nuisance failure indications (indications when no failure was incurred) must be minimized or, if possible, eliminated. Therefore, another ARCS system test effectiveness goal is that the number of false failure indications be less than 5% of the total number of failure indications.

Maintenance actions cannot be considered complete until the installed system is checked to verify that it is operating properly and the aircraft is again available for dispatch. Since this area is of significant importance to the airlines, any automated AFCS maintenance aid must address this aspect of the maintenance process. This consideration becomes of particular importance for an aircraft certified for operation under Category II or III conditions. Following maintenance action, Federal Aviation Administration regulations require that a functional verification test be made before the aircraft can be returned to Cat II or Cat III status. This procedure has typically been a lengthy process, requiring two maintenance technicians and a variety of ground support equipment. Airlines have explicitly expressed a need for an automated system test to perform this verification function with a high level of confidence, where confidence means a high correlation between test results and actual functional integrity.

Operational Constraints.—Quick maintenance action is essential. In many cases, a line maintenance station will not initiate maintenance action for nondispatch modes of an AFCS when turnaround time is less than 4 hours. This current experience reflects excessive time required to isolate faults, replace components, and verify system integrity. It should be possible to conduct system-level maintenance at any line maintenance station that has the required spare LRU's. United Airlines operational experience shows that 25 minutes is available during airplane turnaround for testing and repairing the AFCS. In this time it must be possible to identify the failed LRU, replace the faulted unit, and verify system operational integrity.

To achieve this goal, the airlines press for simplicity in the maintenance operation. Any maintenance action must be based on simple, sound, clear-cut logic that the maintenance crew can readily interpret and respond to. A ground maintenance program requiring the use of ground support equipment (external test gear) generally has a lengthy setup and test procedure that violates the above defined operational requirements. Therefore, the maintenance concept for the ARCS assumes the use of built-in-test equipment only, i.e., ground tests for maintenance purposes, including verification testing, must be performed by line maintenance personnel using only the equipment normally installed in the airplane. It must be designed for operation by line maintenance personnel of a skill level equal to that of today's typical maintenance technician. Furthermore, the ARCS system test concept for routine line maintenance of the ARCS, i.e., maintenance action not involving hydraulic power, is to require only one maintenance technician (two if the test involves hydraulic power).

Any requirement for routine preflight testing is strongly discouraged because of the time and workload burden thus imposed. It is recognized, however, that such testing will be mandatory for certain flight-critical functions. ARCS functions falling into this category are command augmentation and fly-by-wire in the intermediate- and far-term applications, respectively. For these functions, preflight testing will be restricted to only those elements required for dispatch of the aircraft for a revenue departure. Repair of faulted dispatch-required items must be made prior to release of the airplane, while repair of non-dispatch-required items may be deferred. The conduct of a preflight test is the responsibility of the flightcrew and must not require the support of line maintenance personnel.

Maintenance Test Procedures.—Figure 10 illustrates the maintenance cycle as seen by airline maintenance operations. The original detection of a fault condition is normally in flight, manifested by poor system performance and/or fault detecting/annunciation equipment in the system. Ground maintenance action is then initiated as a result of a flight log writeup made by the flightcrew indicating the symptoms of the fault. The flightcrew's responsibility is to report the existence and observed nature of the anomaly; the line maintenance personnel must determine the location of the fault and effect necessary repairs. Before the airplane can be returned to full operational status, the system operational integrity must be verified. Verification testing is primarily to ensure that all required interfaces have been reestablished so that no signal paths are interrupted.

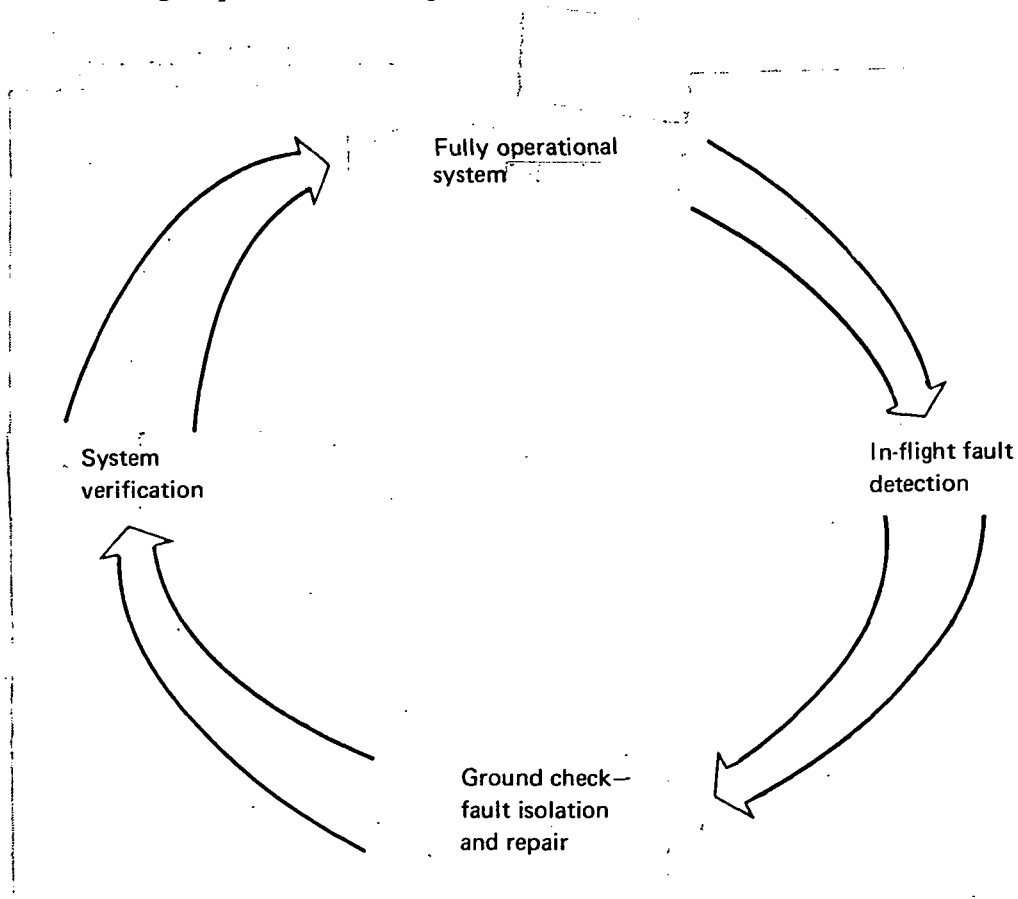


Figure 10.—Maintenance Cycle

In contemporary flight control systems, a major obstacle in a reasonable and systematic system checkout is the complexity imposed when numerous specific preconditions, or pre-test setups, are required for test conduct. Failure to properly set up all required preconditions leads to erroneous test results and hence possible incorrect maintenance action. This condition is believed to be a significant contributor to the unverified removal rate for flight control and associated system LRU's. Though the setup of necessary preconditions is the responsibility of the crew, the ARCS system test is expected to provide an automatic assessment of the status of the required conditions and to alert the operator when they are not properly satisfied.

Certain aspects of system test do not lend themselves to fully automatic implementation. Where such conditions exist, operator intervention with the test process will be requested, together with appropriate instructions via the system test panel (STP) display. Action requests and instructions shall be in the form of an alphanumeric text-type display.

Ground testing shall be broken into three major modules for safe and efficient test sequencing: (1) computer system tests, (2) sensor tests, and (3) servo tests. Transition from computer system testing to sensor system testing shall be automatic, with no operator intervention required. Transition to servo testing, however, must require the test operator to satisfy the servo test preconditions, i.e., hydraulic power, and to ascertain that it is safe to exercise the control surfaces. (This is the reason for having two technicians conducting tests involving hydraulic power. One is required to be outside the aircraft to clear all active surfaces.)

Fault Detection and Recording.—Experience has shown that it is highly possible to have fault conditions that manifest themselves only in flight, due primarily to the dynamic environment which then exists. Since such fault conditions are difficult if not impossible to duplicate on the ground, severe problems are created for the line maintenance operation. The problem is worsened when the flightcrew writeup is unclear or imprecise. The system test design, therefore, must include the capability to automatically detect and isolate failures, particularly sensor failures, while in the dynamic flight environment. Such failure information must be stored for later use by line maintenance personnel so that the need to duplicate the fault condition on the ground is eliminated.

To achieve this capability, the ARCS system test function must provide pertinent failure information relative to in-flight fault conditions that is accessible for display as a part of the ground maintenance operation. This failure information must supply the line maintenance technician with sufficient information to allow him to localize the fault condition to a specific LRU. These considerations led to the establishment of an ARCS system test requirement that a section of nonvolatile memory be provided for storage of this data. Though some historical record of past failure history may be beneficial to engineering and/or shop maintenance, the only failure data that need be provided for line maintenance is that pertaining to the immediately preceding flight segment.

In addition to in-flight fault detection/recording capability, a means must be provided whereby a ground crew can initiate an automated system checkout sequence as part of the ground maintenance operation. This capability will assist ground maintenance by locating permanent faults within the system, by testing for potentially latent failure conditions not otherwise detectable by on-line monitors, and by verifying system operational integrity following maintenance action.

When required, a preflight test procedure shall be provided that can be viewed as a subset of the total ground test procedure. Failure storage memory shall be provided for recording the results of this test procedure for display to the test operator.

All failure data storage, whether accumulated as a result of in-flight testing, operation monitoring, or ground testing, shall be recorded in a manner to explicitly identify the failure to an LRU level. Airline operators have asked that the failed LRU be identified by name in an alphanumeric format, wherever possible. In this way potential ambiguities due to misinterpretation can be eliminated.

System Test/Crew Interface. A means must be provided for interfacing the system test function with the test operator, either the flightcrew or ground maintenance crew. This interface shall be provided by a system test panel (STP) encompassing all that is necessary for the operator to initiate the test function, control its progress, interact where required, and display test results.

The system test panel is intended to be used by the flightcrew for preflight test and for in-flight display of system operational status and by the ground maintenance crew for maintenance-level testing. Since the flightcrew requires access to the panel functions and ground testing involves the use of certain flight deck controls, the STP must be designed for installation in the flight deck where it is accessible for both functions.

Insofar as possible, the STP shall be designed so that the ARCS appears to an operator as one system. It shall be used to enunciate LRU-level failure information, system status or operational capability, and operator action requirements in an alphanumeric format.

A distinctly different type of information is needed by the flightcrew compared to that required by ground maintenance. The flightcrew need only know of the existence of a failure condition if it affects the minimum operational capability (flight-critical mode) or the currently selected operating mode. The maintenance crew, on the other hand, must know of the existence of the malfunction and its location.

4.1.2 FLIGHT SAFETY CONSIDERATIONS

Specifying safety-of-flight criteria and goals implies defining the level of accident risk, or probability of accident, that can be tolerated for the total aircraft and the risk contribution that can be tolerated because of particular subsystems in the aircraft.

The safety-of-flight criteria and goals, or functional reliability criteria and goals, defined for the ARCS were developed from three considerations. Two of these, and the basis for the criteria, are the U.S. and British airworthiness regulations or proposed regulations. Where interpretations of these regulations were necessary for a specific application, or when additional criteria were needed to completely specify an operational risk situation, an independent assessment of the involved factors was made. Since the criteria and goals thus identified will be of no use unless systems can be economically implemented and operated in a manner to meet these criteria, the third consideration was the implementation of projected realism integrated into the criteria definition process.

The paths followed in defining safety-of-flight criteria for the CCV/FBW function and the autoland function are illustrated by figure 11. Starting points are the U.S. Federal Aviation Regulation (FAR) requirements and certain proposed British Civil Airworthiness Regulation (BCAR) requirements pertaining to flight control systems. The result of the definition process is a set of suggested criteria addressing the average functional reliability required for the CCV/FBW and autoland functions and the specific risk associated with the CCV/FBW function on a long flight (10 hours). In the process, current certification practice in the definition of extremely improbable events, the definition of hazardous events, and the actual current jet landing accident rate are discussed.

4.1.2.1 CCV/FBW Functional Reliability

The general regulatory safety requirement that governs the design of flight control systems and other equipment, systems, and installations in commercial transports is stated in FAR Part 25, paragraph 25.1309(b), dated August 5, 1970.

The airplane systems and associated components, considered separately and in relation to other systems, must be designed so that—

- 1) *The occurrence of any failure condition which would prevent the continued safe flight and landing of the airplane is extremely improbable, and*
- 2) *The occurrence of any other failure conditions which would result in injury to the occupants, or reduce the capability of the airplane or the ability of the crew to cope with adverse operating conditions is improbable.*

Failure conditions shall include considerations of electrical and hydraulic power loss, including loss of one engine on two- or three-engine airplanes, and two engines on four- or more-engine airplanes.

Although not expressed in regulatory documents, a number less than or equal to 1×10^{-9} has been imposed upon manufacturers by the FAA in recent aircraft certification programs to represent the probability of an event designated as extremely improbable. An improbable event by FAA definition is one expected to occur with a probability between 1×10^{-5} and 1×10^{-9} .

Insight into how to interpret and apply the “extremely improbable” concept is furnished by the following excerpt from reference 1, a paper presented by an FAA official:

There have been many careless and inaccurate references to a probability of “ten to the minus nine,” giving a vague and misleading impression that any failure condition which is to be accounted for at all must be shown to be extremely improbable: that is, shown by analysis that the probability of occurrence of such a failure condition is less than 10^{-9} .

This is only true if the effect of the particular failure condition being considered would preclude any possibility of continued safe flight and landing of the aircraft: or which, in a word, is catastrophic.

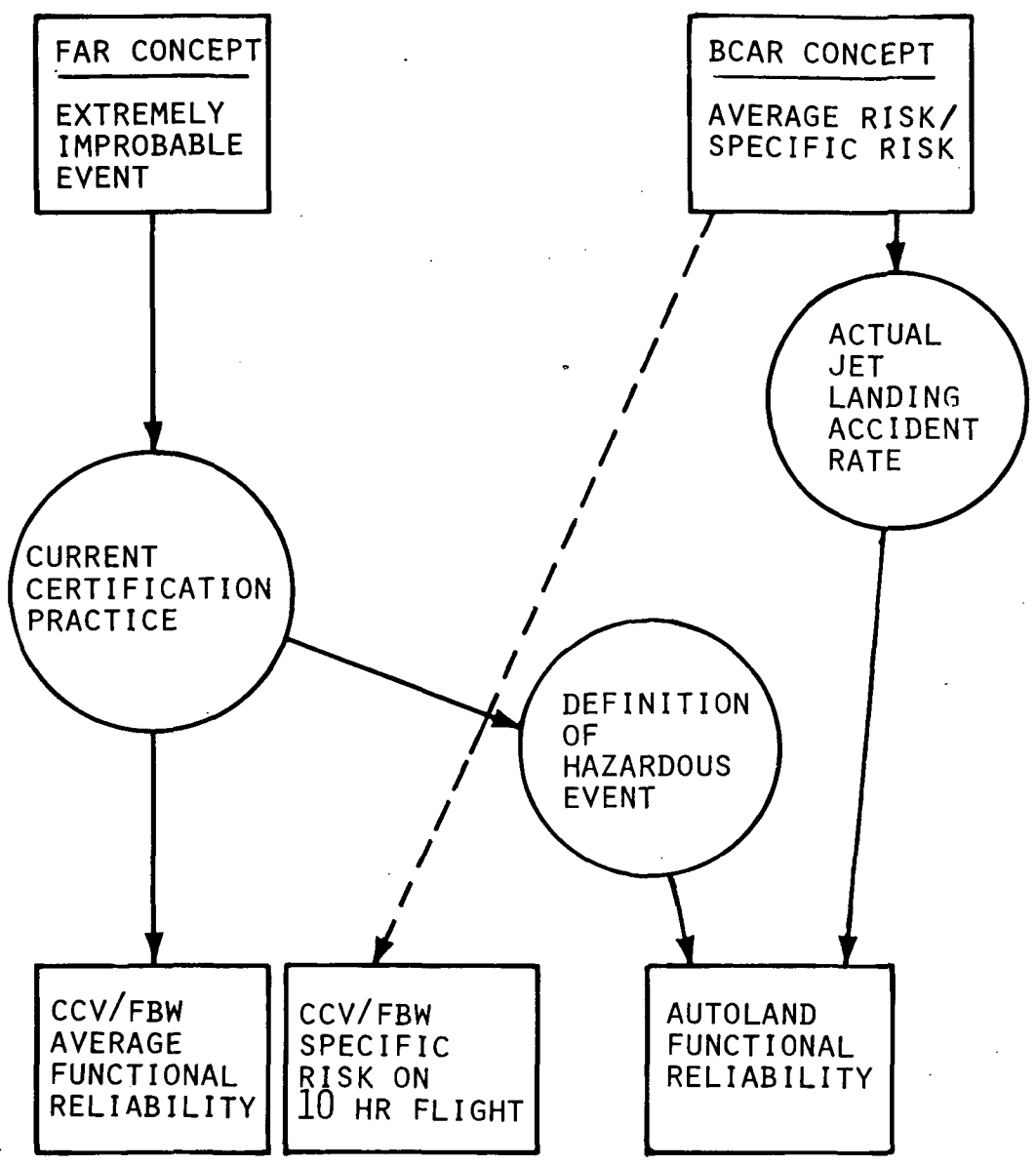


Figure 11.—ARCS Functional Reliability Criteria Definition Process

The significance of this statement is that if it can be shown that the loss of the CCV/FBW function is extremely improbable, the functional reliability requirement is satisfied. In other words, it is *not* required that all catastrophic failure conditions (loss of FBW, wing spar failure, etc.) taken together be extremely improbable.

With the background presented above, and taking a Boeing conservative position that all CCV/FBW function failures are catastrophic, the formulation of the first CCV/FBW criterion becomes straightforward:

Loss of the CCV/FBW function, given a fault-free system at dispatch, shall be extremely improbable.

This requirement addresses the average risk (fatal accident probability) over all flights in all aircraft using the system considered. The average risk is the combined effect of the flight duration and the system failure probability, which increases with exposure time as illustrated by figure 12.

For the fly-by-wire application, a mission shall be assumed to start at the time of dispatch and end at the time the aircraft leaves the runway after landing at the destination or at an alternate airport. Mission times of up to 10 hours shall be considered, and an average flight time of 1 hour per flight shall be assumed (based on United Airlines data).

In addition to the average risk, it is desirable to specify a maximum ceiling on the specific risk associated with a long flight. A proposed BCAR (ref. 2) suggests that the specific risk associated with an autoland be no higher than the risk of another flight, i.e., the average actual in-service accident probability for commercial transports. Adapting this concept to the cruise situation would yield (again postulating a maximum flight time of 10 hours): the specific risk from all causes on any flight of up to 10 hours shall be lower than the total current accident risk.

The total accident data worldwide during 1968-1973 shows approximately four fatal accidents per 1 million flights. The average flight time was slightly over 1 hour. This data was extracted from NTSB published accident reports.

Allowing a 5% budget for FBW-related accident causes—the proportion of causes attributed to flight control systems according to the 1968-1973 accident data—a second requirement for the CCV/FBW function was formulated:

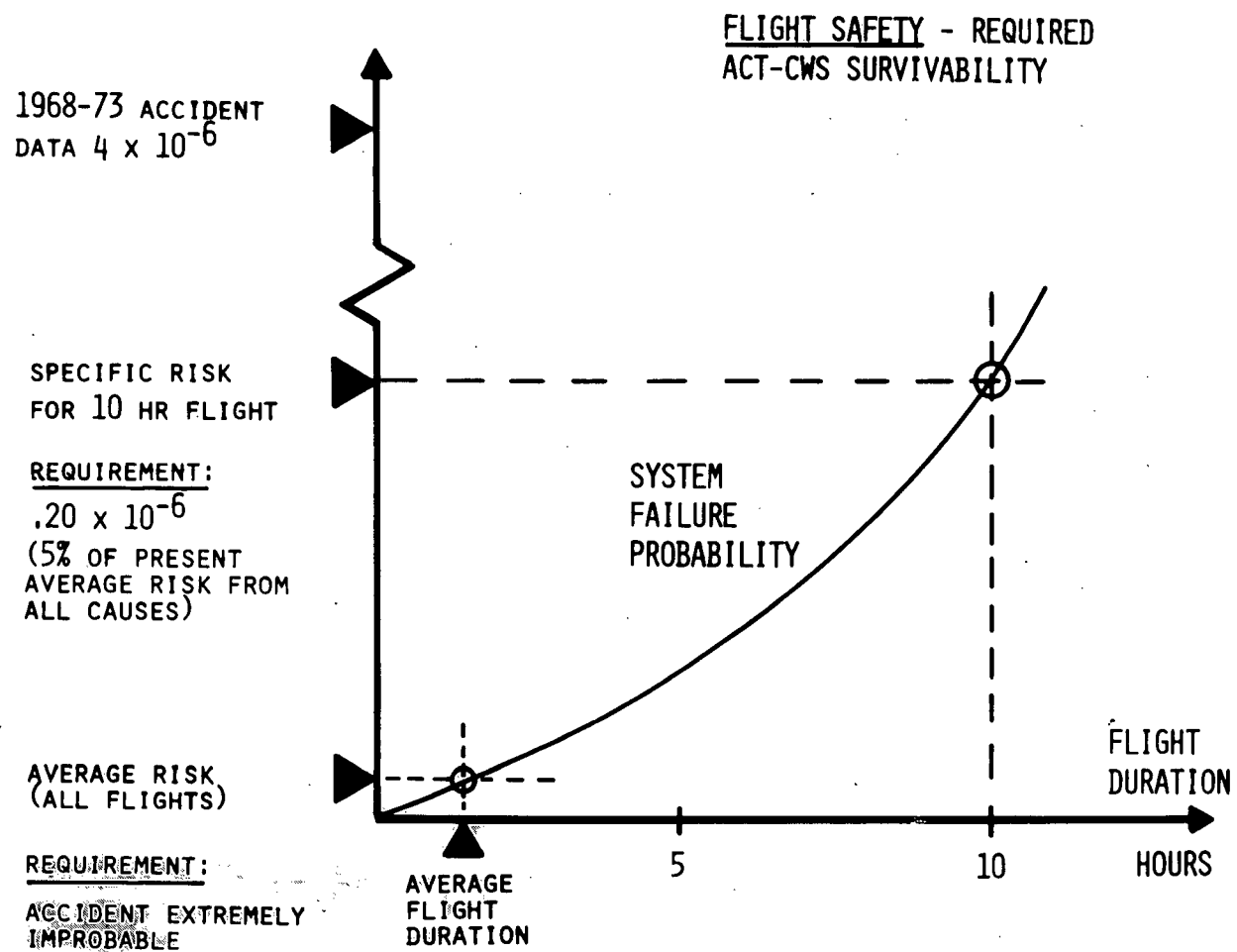


Figure 12.—CCV/FBW Safety-of-Flight Criteria

The specific risk contributed by the CCV/FBW function on any flight of up to 10 hours' duration shall be less than 0.20×10^{-6} .

In addition to the average risk and specific risk, which apply to the normal operation of the CCV/FBW function, the particular operational circumstances and reliability requirements associated with a functional failure state leading to the necessity of a diversion from the intended flight plan must be specified.

The pilot must be notified if a failure state occurs that significantly reduces the CCV/FBW functional survival probability below normal levels. This failure annunciation must occur such that the overall functional reliability requirement is not violated. The total CCV/FBW failure probability $P(\text{FBW fail})$ can be expressed as a conditional probability

$$P(\text{FBW fail}) = P(\text{FBW fail/diversion}) \cdot P(\text{diversion})$$

Where $P(\text{diversion})$ is the probability of diversion. The third CCV/FBW safety-of-flight requirement then becomes:

The system shall annunciate when a failure state requiring diversion has been incurred so that, including an additional 30 minutes of flight, the combined event of incurring a diversion and a system failure after incurring the diversion is extremely improbable.

The 30-minute time period following a diversion decision was specified for the U.S. SST.

4.1.2.2 Autoland Functional Reliability

Although automatic landing is not the exclusive method allowed by regulations to achieve Category III weather capability, it is the primary one pursued at present. This indicates the difficulties associated with low visibility and the human pilot's ability to cope with restricted visual cues during landing. Use of autoland in good visibility—including Category II weather conditions and better—is strongly supported by segments of the airline industry to improve overall safety.

Specific regulatory criteria for Category IIIa landing systems are expressed by FAA Advisory Circular (AC) 120-28A, Section 3, dated December 14, 1971:

Operational Concepts—The total airborne system must be designed and must provide sufficient information to the pilot so that the landing may be safely continued and completed or a go-around safely executed from any altitude following any single failure or combinations of failures not shown to be extremely improbable.

A crucial issue in the probabilistic approach is the definition of the event to which the probability is attached. For the ARCS program, the conservative assumption was made that the pilot will not intervene and recover from a system failure that occurs below the "alert" height in Category III conditions. The alert height is aircraft dependent, but typically between 50 to 100 feet above the runway.

When the autoland function is used during conditions that provide runway visibility above Category III minima, the pilot will be credited with at least a 0.99 probability of recovering from any autoland system failure below decision height and either safely complete the landing or execute a go-around. This assumption has previously been used in certification context.

A system failure during autoland is catastrophic only under two circumstances: (1) the visibility is such that the crew cannot recover from the failure effect or (2) the crew fails to recover despite favorable visibility conditions. More strictly formulated, the hazardous events in connection with autoland can be expressed as:

1. Autoland failure given Category III conditions
2. Autoland failure given pilot failure to recover in non-Category III conditions

The hazardous exposure lasts for approximately 45 seconds (from decision height to stop on the runway).

AC 120-28A requires that each case of hazardous events be extremely improbable. Accepting for a moment that extremely improbable implies a probability of occurrence equal to or lower than 10^{-9} , items 1 and 2 can be written:

3. $P(\text{autoland failure}) \cdot P(\text{Cat III}) \leq 10^{-9}$
4. $P(\text{autoland failure}) \cdot P(\text{pilot fail}) \leq 10^{-9}$

The probability of incurring Category III weather in the U.S. is approximately 0.5%. Assuming that this probability worldwide is less than 1% and recalling the assumption that the pilot will recover from at least 99% of all autoland malfunctions in non-Category III conditions, items 3 and 4 yield the same result:

$$P(\text{autoland failure}) \leq 1 \times 10^{-7}$$

Therefore, for the autoland function, a failure probability lower than 1×10^{-7} for a 45-second exposure time must be achieved to meet the FAA requirement.

The corresponding BCAR requirement for autoland average risk is formulated as follows:

The system shall be such that the total fatal landing accident rate (i.e., average risk) due to the use of the system at any time and in the new visibility conditions permitted below current minima (approx. 200 ft and 1/2 mile) shall not be greater than the present total fatal landing accident rate for all transport aircraft. This figure is believed to be of the order of one fatal accident per million landings. Since piloting is only one of several possible causes of fatal landing accidents, the system should not contribute a rate greater than 1.0×10^{-7} fatal accident per landing.

The actual total fatal landing accident rate for worldwide jet transport operations during 1968-1973 was approximately 2×10^{-6} .

The BCAR concept deals with the *use of the system*, which implies two potential causes for a hazard: risk due to system malfunction and risk due to performance. If 10% of the landing risk from all causes is allocated to piloting and this 10% is split equally between autoland malfunction and performance, a resulting required probability of malfunction lower than 1×10^{-7} is obtained.

Thus, the functional reliability requirement established to satisfy the FAR criterion equally satisfies the BCAR criterion.

4.2 ARCS DESIGN REQUIREMENTS

The operational requirements identified in the previous sections need to be translated, interpreted, or expanded to take a form in which they can serve as design requirements for the ARCS. The most significant aspect to recognize in this translation process is that the computer system is an integral part of a total flight control system; it must be designed and optimized in that context.

The ARCS development takes off from the state of digital flight controls technology achieved to date. The GE MCP-703 (WWCS) system, which is the baseline system for the ARCS, represents the third generation of flightworthy experimental digital flight control computers, developed and tested extensively. The ARCS design requirements formulated below have evolved from this development experience. They purport to define the most desirable direction in which to pursue a fault-tolerant computer system for a critical flight control application on a transport aircraft. The three ARCS design requirements—system functional design, software design, and hardware design—are shown in figure 13 and are discussed further in the following sections.

4.2.1 SYSTEM FUNCTIONAL DESIGN

The basic design objective for the ARCS was to develop a highly fault-tolerant, redundant channel system for application in an airborne environment. The fault tolerance includes the ability to withstand transient fault conditions by temporarily degrading to a lower redundancy level, with recovery of redundancy when the transient fault condition disappears, and the ability to remain operational after two like component failures by degrading from triplex to duplex to simplex.

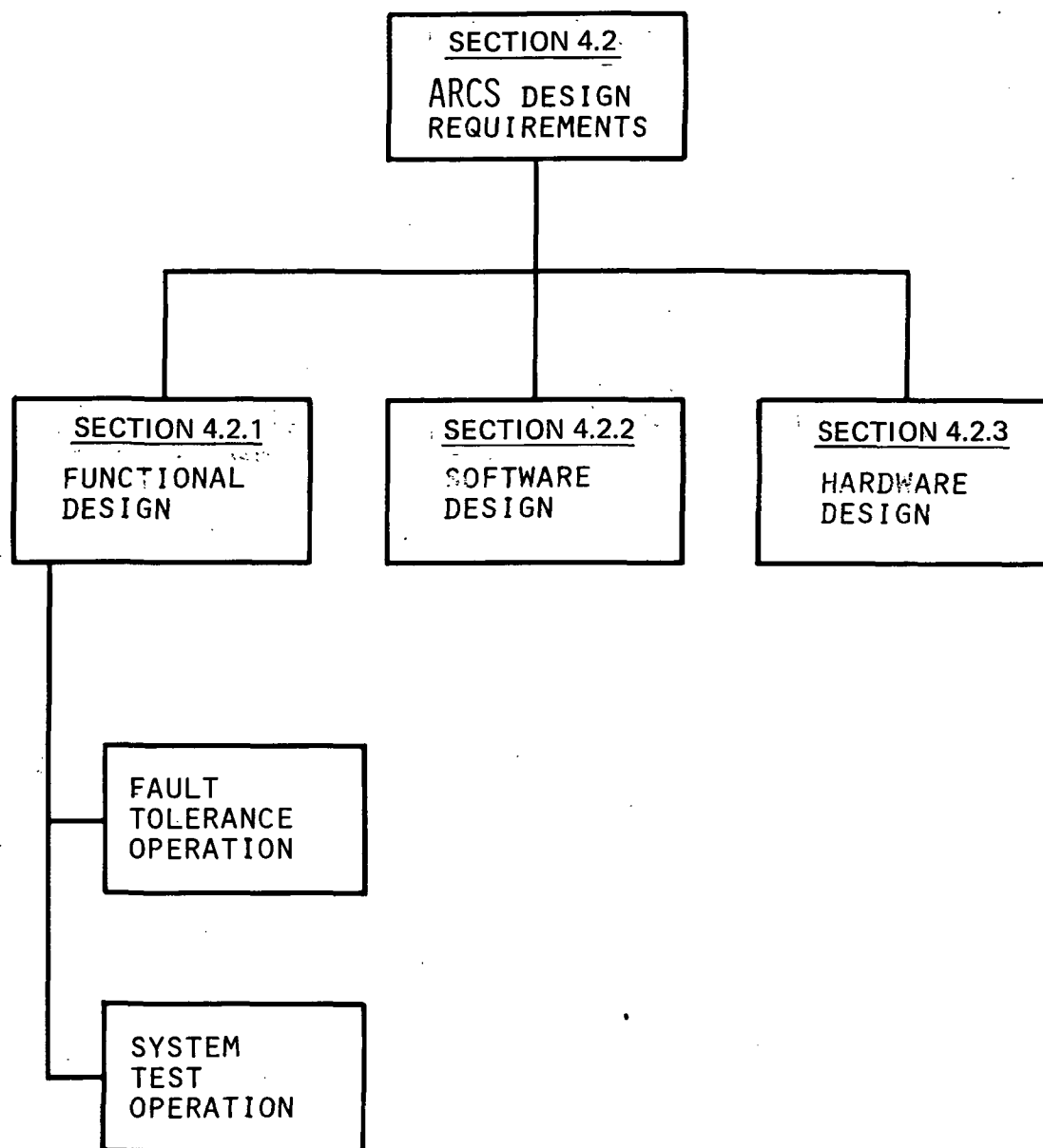


Figure 13.—Elements of ARCS Design Requirements

In the past, fault-tolerant analog automatic flight control systems have been designed to survive one (triple-channel fail-operational) or two like (quad-channel two-fail-operational) failures. A required effort included in the design task was to prove that any conceivable like component failure beyond the first or the second failure, respectively, would result in a safe system shutdown.

Because of the nature of the digital system, with timesharing of hardware and multilevels of dependency between elements of the system, the relatively simple analysis used for redundant analog systems is no longer realistic. Instead, a probabilistic assessment approach must be taken to design integrity into the system and to analyze the resultant design. This probabilistic approach must be reflected in the formulation of design requirements.

A significant distinction exists between the traditional and the present design problem: The purpose of the ARCS design is not to achieve strictly two-fail-operational fault tolerance (in a triplex configuration), but to achieve a low probability of system failure. In other words, a system failure after only two like failures is acceptable provided the probability of that occurring is sufficiently low.

The discussion of ARCS requirements rests on a set of key terms, defined in section 4.2.1.1. The design requirements for the fault-tolerant operation are covered in section 4.2.1.2. System test, a crucial aspect with respect to ensuring that the basis for the probabilistically derived conclusions on system reliability and safety indeed exists at crucial points during system operation, is treated in section 4.2.1.3.

4.2.1.1 ARCS Key Definitions

The total flight control system is comprised of *modules*, or sets of elements, performing a specified function. The system is organized vertically into *stages*, or sets of redundant modules, and horizontally into channels. A *channel* is a unique set of modules, which together are capable of performing the system function.

The ARCS includes modules such as the processor, memory, input/output, power supply, and system test panel. Modules interfacing with the ARCS are sensors, mode select panel, servos, and displays.

A *fault* is defined as a performance anomaly. A *transient fault* is a temporary performance anomaly, while a *failure* is a permanent fault.

Tolerance is the ability of the system (or stage) to continue to perform the required function given a fault. *Coverage* is the conditional probability that, given a failure, the system (or stage) continues to perform the required function.

Reconfiguration is the process of attempting to tolerate a fault. This process has two possible outcomes: recovery or redundancy degradation. *Recovery* is the reestablishment of the operational level of redundancy that existed prior to the occurrence of a transient fault. *Redundancy degradation* is the reduction of the operational redundancy level of a stage.

A fault whose presence can be detected by implemented self-testing and/or monitoring procedures is called a detectable fault. The discovery of the existence of a fault is termed *detection*. A *latent failure* is a failure that has not been detected.

Localization is the identification of the particular module in which a detected fault has occurred, and *isolation* is the setting apart of a faulty module in such a way that it cannot affect the system output.

4.2.1.2 Fault-Tolerant Operation

The basic design objective for the ARCS computer system is to minimize hardware redundancy yet maximize fault tolerance in the event of (1) computer system, sensor system, and servo system component failures and transient faults and (2) other transient interruptions that influence the system operation in the airplane operational environment.

The first major assumption made for the ARCS definition study is that one basic organization, or architecture, for the electronic flight control system can be developed to effectively handle the full spectrum of presently envisioned functions. Based on existing experience in fault-tolerant flight control system design, it is assumed that a successful candidate ARCS will exchange sensor information between channels to achieve maximum system reliability and that the system will utilize cross-strapped computer information for redundancy management and reconfiguration purposes. The following ground rules and requirements anticipate such a system.

The most disastrous form of a failure mode in an ARCS would be where channel dependence could be a source of a single-point failure that could cause system failure. It is therefore of critical importance to guarantee channel independence in the probabilistic sense of "independent events." The first three requirements below address this particular concern.

1. *Channel self-dependence*—System and channel status must be registered in each channel in order to have the potential for achieving fail-operation from dual to single and reconfiguration back to higher orders of redundancy.

Requirement: Each computer shall independently assess its own operational status.

2. *Channel integrity*—No single-point hardware failure and no software routine in one computer shall have the potential to interrupt the operation of another computer by bypassing the other computer's decision-making routines.

Requirement: No computer operation, or combination of computer operations, shall interrupt the normal operation of another computer.

3. *Fault propagation*—Failures shall not have a potential to propagate across channels.

Requirement: No servo shall be controlled by processes outside its own channel.

High failure coverages are essential to achieve the system reliability demanded by the operational requirement. A first-failure coverage of unity is deemed a realistic design goal. The following items address the fault tolerance of the system.

4. *Output voting*—Based on considerable experience in the technology of mechanical servo output voting, the following rule was adopted.

Ground Rule: A mechanical servo output voting node, providing a system coverage of unity for any first-failure condition in a triplex configuration, shall be assumed for the ARCS development.

5. *Second-failure coverage*—Computer self-test and wraparound test have the potential of providing a thorough fault-detection capability.

Requirement: A second-failure module coverage of 0.95 or better is a design goal for the computer and interface modules.

6. *Simplex-failure coverage*—Coverage of a failure occurring in simplex operation will be of importance in decreasing the probability of an active system failure.

Requirement: A simplex failure-detection probability of 0.90 is a design goal for the computer and interface modules.

7. *Sensor signal selection and failure detection*—The sensor selection algorithm must be able to isolate the effects of all types of faults, including open failures, hardover failures, and any ramp failures and oscillatory failures.

Requirement: The signal selection algorithm must provide an output that is acceptable to the application task for normal (unfailed) operation and during any fault condition in one of the redundant input signals.

Requirement: Means must exist to detect and isolate the effect of a sensor failure, as well as to revise the failure-detection algorithm to be compatible with the lower redundancy, before the probability of incurring an additional like failure may be high enough to create an unacceptable risk.

Requirement: The failure-detection algorithm must operate in the presence of normal signal tolerances such as biases, scale factors, and linearity errors, and in the presence of noise.

Requirement: The proportion of “nuisance failures” (leaky transients) due to signal tolerance and noise must be insignificant compared to the number of genuine failures.

8. *Automatic start and restart*—The system must be designed with a capability to recover from massive transient faults, which could be caused by phenomena such as lightning strokes and which can result in simultaneous disagreement between all voted or monitored signals.

Requirement: The system shall be designed with the capability of automatic start and synchronization after power turn-on, as well as automatic restart and resynchronization after transient power faults or massive transient signal faults.

9. *Pilot intervention*—Transient fault conditions are a major source of system degradation in today's redundant systems. Tolerance against transient fault effects is a major design requirement for the ARCS.

Requirement: Pilot intervention shall not be relied upon for system reconfiguration processes.

Requirement: System start, restart, and synchronization shall not require pilot intervention.

10. *Allowable recovery time*—The time allowed for reconfiguration from duplex to simplex before declaring system failure is a significant parameter for the fault-tolerant computer system. This time is dependent on the particular vehicle application. For the ARCS functional success analysis a time period longer than 1 second between fault occurrence and successful system reconfiguration was selected as the criterion of system failure. It is judged that vehicle control can be recovered if a critical function is restored within this time period.

Requirement: The system must return to successful reconfiguration within 1 second after fault occurrence.

4.2.1.3 System Test Operation

System test encompasses functions necessary to ensure the integrity of the fault-tolerant system. Reliability predictions, on which the operational risks during the use of the system are based, rest on assumptions about the failure status of the system at any particular time. These assumptions must be validated continuously through a thorough verification that all failures in the system are known as soon as possible after they occur.

It follows from the close relationship between finding a failure and effecting its repair that system test for the purpose of maintenance action is inherently part of the same problem complex. System test is therefore considered as one function with two purposes: integrity assurance and maintainability enhancement.

The ARCS system test function shall encompass ground test, in-flight test, and in-flight monitoring with the following specific purposes:

- The ground test function shall be organized as three distinct phases: (1) a self-test of the computer subsystem, (2) a test of the sensor systems, and (3) a servo subsystems test. Sensor and servos shall be tested both statically and dynamically wherever possible. Any preflight testing of the ARCS shall be a subset of the ground test function.
- In-flight tests shall be conducted where necessary to cover subsystem failures that may not be detectable by existing system monitors.
- In-flight monitoring shall be used whenever similar data are available from separate independent sources to detect and isolate failures. When a sensor stage that is less than triply redundant is monitored, the technique of "pseudo-sensor monitoring" shall be used to augment the normal cross-channel comparison. Pseudo-sensor monitoring refers to the practice of deriving sensor data from a related but not identical source.

The system test function shall be self-contained in each channel so that it is operable regardless of the number of operational channels. Control of the function and display of results shall be the same whether the system is triplex, duplex, or simplex. The control and display panel (CDP) shall interface with the computer in each channel.

Manual initiation of the test function shall be prohibited in flight in order to prevent an inadvertent activation of a test that would disrupt the normal system operation. The only manual input to be acknowledged in flight would be a read request that would display the current operational or failure status of the system.

The ARCS system test function shall include the total flight control system in the following testing sequence:

Computer system test
Input and output test
Sensor tests
Servo tests

Computer system test—As part of this test, cross-channel data links and independent computer monitors shall be tested. A processor self-test program shall verify proper operation of each computer instruction by testing each microstate. The operation of the processor clock shall be continually monitored. All data being read out of variable memory shall be tested for correct parity. The occupied portion of the variable memory shall be sequentially read out to test for invalid parity. The contents of each location in the nonvariable (program) memory shall be added together and compared with a prestored value to assess the status of the memory.

Input/output tests and sensor tests—Wraparound loop tests shall be used to verify the multiplexed portions of analog and discrete input/output signal paths. Sensor self-test functions shall be exercised as a part of the ground tests to verify basic sensor availability and interface integrity.

Servo tests—Ground testing of the servo systems shall verify the engagement/disengagement function, test the dynamic response, and verify the force voting characteristics.

4.2.2 SOFTWARE DESIGN

The operational software is an element of the fault-tolerant computer system that is common to all the redundant channels. A software error will not cause a disagreement between redundant channels, will therefore not be detectable by the system, and thus constitutes a potential single cause of system failure.

In principle, there is no difference between hardware design/mechanization errors and software design/implementation errors. The only possible way to eliminate such errors is by careful design, clearly defined mechanization or implementation standards, and testing.

Reliable (correct) software is an obvious requirement for a fault-tolerant computer system. Reliable software requires a strict software design methodology. The software design methodology adopted for the ARCS design process must recognize that the single most important factor in achieving reliable software is to maintain design and implementation visibility on all levels throughout the software development process. This visibility is necessary to ensure that all requirements are being correctly interpreted and met and to provide continuity across interfaces between personnel involved in the development, testing, certification, and operational maintenance of the system. The relationships between these processes are illustrated in figure 14.

The first interface area—software design—is illustrated in figure 15. The software designer will develop the software requirements in cooperation with the control system engineer, who is responsible for the specification of the overall system and its performance. After the software design is complete, the designer must present his design in a format that will serve three purposes: it will be the input to the programmer for implementation into code, it will be an input to test planning for the verification and validation process, and it will provide feedback to the control system designer.

The second area of interface, illustrated in figure 16, concerns the certification and operational processes. The engineering representatives of the certifying authority will interface with the system and software designers during the airworthiness validation phase of a new system's introduction on a commercial transport. Finally, the operator's Engineering department will need to know the design to a certain level in order to maintain the equipment.

The software design methodology used to describe the ARCS design must provide the visibility needed for the above purposes. It must result in unambiguous, rigid software definitions to ensure high confidence in the integrity of the fault-tolerant system and to provide design visibility necessary for certification of flight-safety-critical functions implemented in, and described by software.

4.2.3 HARDWARE DESIGN

In addition to satisfying the environmental design requirements specified for airborne avionics in reference 3, the following specific ARCS or flight-controls-oriented requirements must be met.

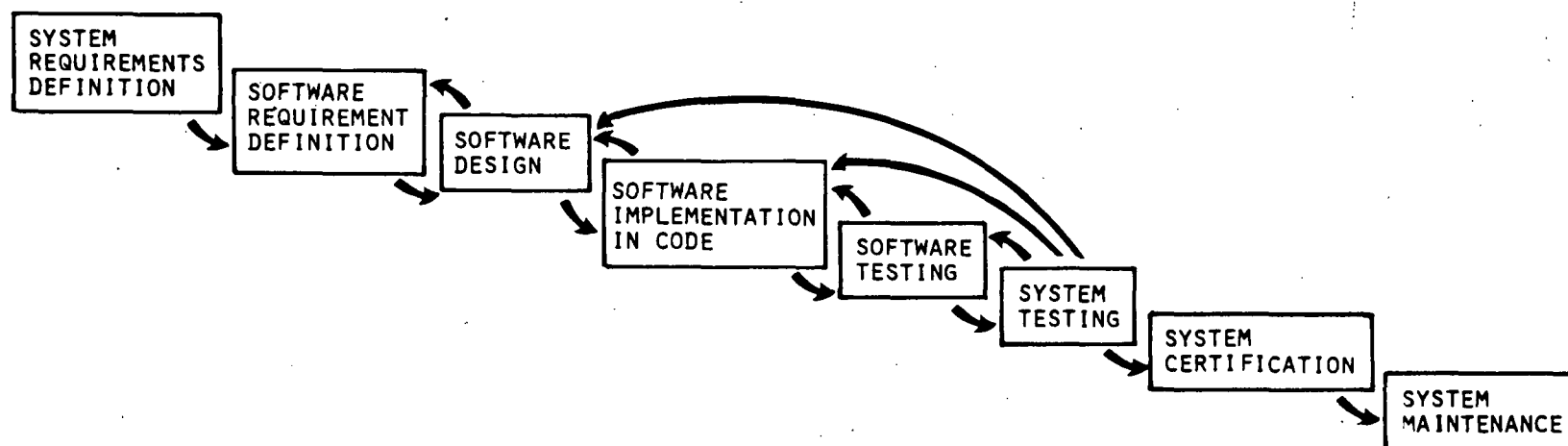


Figure 14.—Software System Development Process

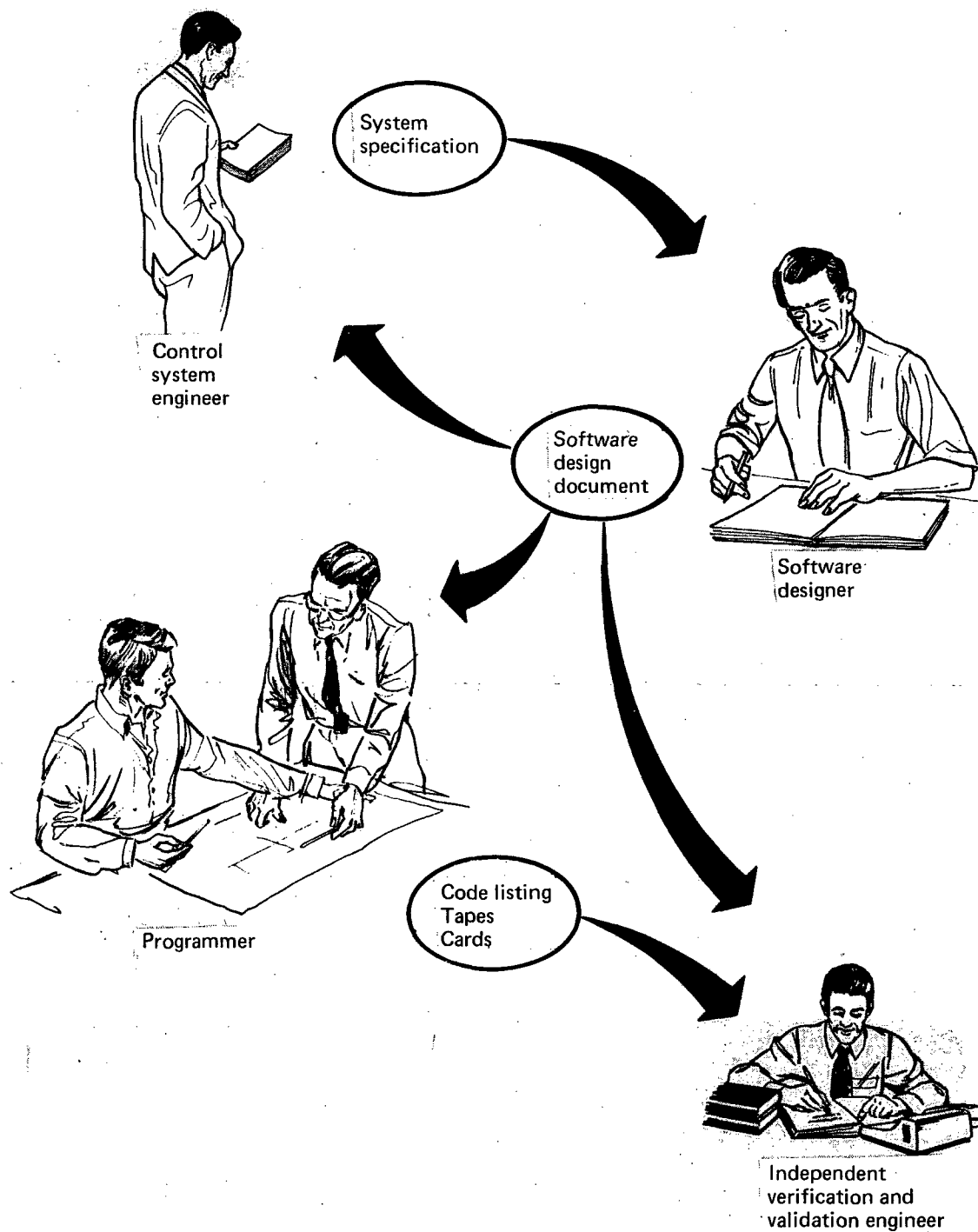


Figure 15.—Software Design Interfaces

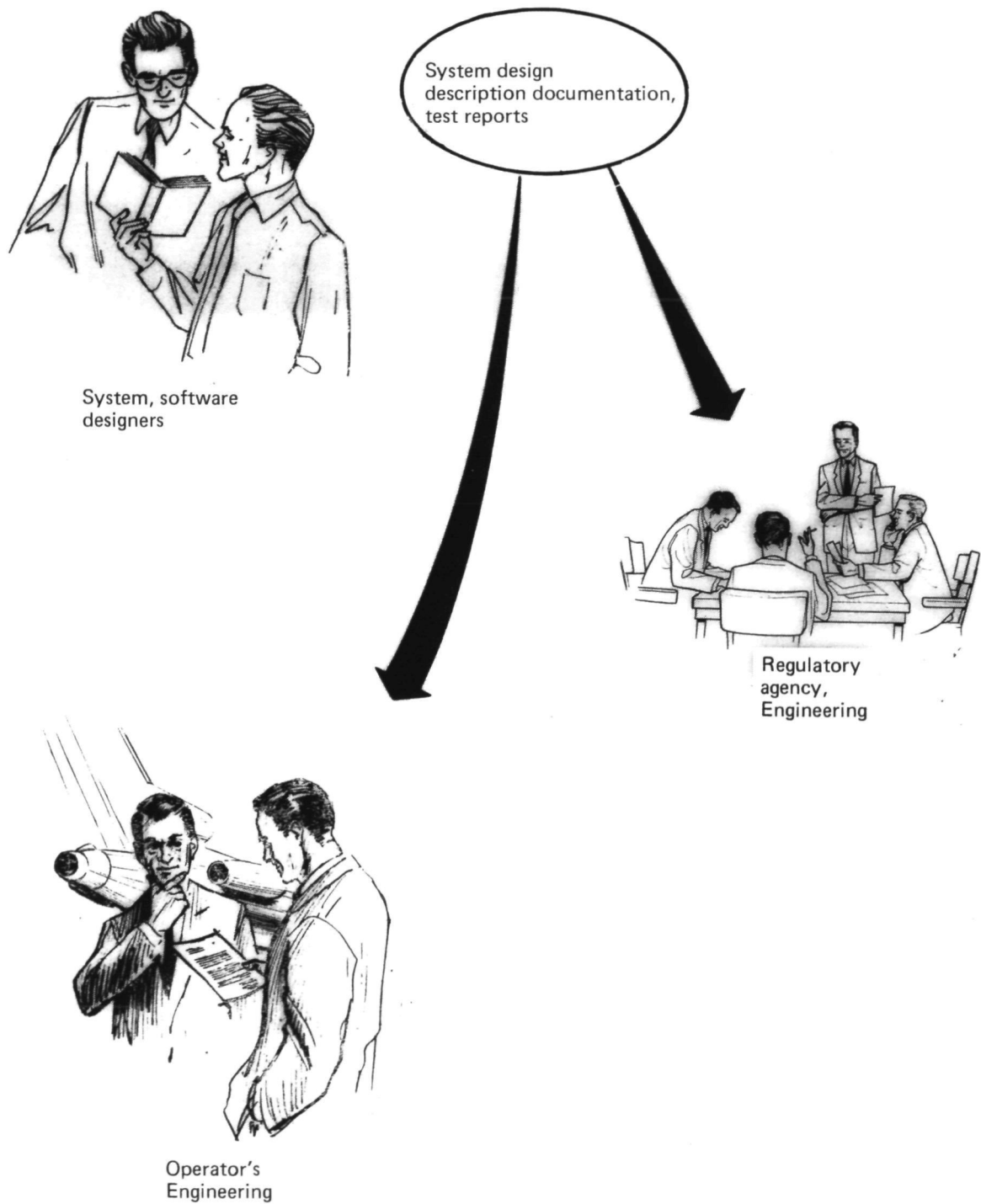


Figure 16.—Certification and Operation Interfaces

4.2.3.1 General Architecture

The final detail configuration of flight control functions can be established only through flight testing. For an efficient flight test program requiring a minimum turnaround time for software changes and the potential for making in-flight software changes, the computer system must be capable of working with an electrically alterable memory such as a core memory.

For the fully certified production-type system, on the other hand, a completely nonvolatile program memory is anticipated to achieve the highest possible integrity and fault tolerance in the system. Therefore, the processor-to-memory interface for instructions and constants storage shall facilitate both core and ROM types of memory.

The computer system shall be applicable to flight control systems performing functions of various mixtures of flight criticality and mission lengths, resulting in different requirements for redundancy. The ARCS hardware shall therefore be adaptable to at least quadruplex redundancy without requiring any changes to the architectural design.

An initial memory size estimation of 16K has been identified to implement the software system for the ARCS far-term application model. To provide a sufficient growth margin, the memory shall be expandable to a minimum of 32K.

The word length required for flight control computations is 16 bits. For strap-down inertial navigation, 24-bit computations are required for certain calculations. Because of the potential of applying the ARCS computer to strap-down computation, a basic word length of 16 bits with the capability of efficient double precision, or a basic word length of 24 bits, is required.

4.2.3.2 Processing Speed

The far-term application model for the ARCS requires an estimated 425 000 operations per second (425 kops). To provide a sufficient margin for development and growth, a computational capability of 600 kops shall be the goal for the far-term application model. The processor timing shall allow minor cycle times down to 10 ms to satisfy the structural mode control requirements of the far-term application model. As a preliminary requirement for transient fault recovery, a cross-channel data transfer rate of 50 words/ms shall be anticipated.

4.2.3.3 Design Integrity

The ARCS shall be designed to withstand and suppress the effects of electrical hazards including lightning stroke. The ARCS hardware design shall be constructed to survive a cable failure as a single failure, where all the wires in a cable may be open-end or shorted to ground. It shall also survive, as a single failure, the computer LRU sliding out of the receiving tray, where all pins on the LRU connectors disengage almost simultaneously.

5.0 ARCS DESIGN CONCEPT

The ARCS was designed from criteria and requirements spelled out in section 4. It is convenient to describe the resulting design from three distinct aspects: functional, software, and hardware. This yields a top-down definition of the system concept as indicated by figure 17. The functional description is the top level—a direct expansion of the design requirements into functional concepts. From these functional concepts (sec. 5.1), the design is described in progressively increasing detail as it is implemented in software and hardware. The software design (sec. 5.2) was pursued to a stage from which coding can be initiated, whereas the hardware design (sec. 5.3) stops short of the final detail design definition required for hardware fabrication.

5.1 FUNCTIONAL DESCRIPTION

On the highest functional level, the ARCS performs four distinct tasks. The first and foremost, since it is the ultimate justification for the system, is the application task of controlling the aircraft. The design of aircraft control functions is not, however, the object of the ARCS development, although performance of those functions was a consideration integrated into all ARCS design decisions.

The second task of ARCS, and the primary forcing function for the fault-tolerant design, is that of providing maximum functional survivability in the presence of transient and permanent fault conditions. Inherent in the fault tolerance is also the third task, namely to ensure system integrity with respect to flight safety. Discussion of those elements that comprise the fault-tolerant aspects of the ARCS design make up the bulk of the functional description.

The fourth task of the ARCS is that of fault status verification. Because of the maintenance aspects and their significance to airline operations, a system test function was developed as an integral element in the ARCS design. However, certain aspects of fault monitoring and system failure status are inherently a part of the system integrity complex. System test is therefore, by necessity, integrated with the fault-tolerant design aspects of the ARCS.

5.1.1 SYSTEM RECONFIGURATION

Reconfiguration has been defined as the process of attempting to tolerate a fault. In dealing with the various fault conditions that can cause a transition between possible states in the reconfigurable system, the process of power-up was considered as one of the reconfiguration processes. In so doing, a simple, straightforward reconfiguration strategy was adopted to handle all such transitions. In this strategy, power-up of a second channel is identical to a recovery from a transient fault in duplex operation, and power-up of a third is equivalent to a recovery in triplex.

The general reconfiguration process for the total system involves one or more of the sub-processes of fault detection, fault localization, fault isolation, recovery, and redundancy degradation. Figure 18 shows the possible outcomes as a result of faults occurring in the system. Transient faults will result in restoration of the operational state that prevailed

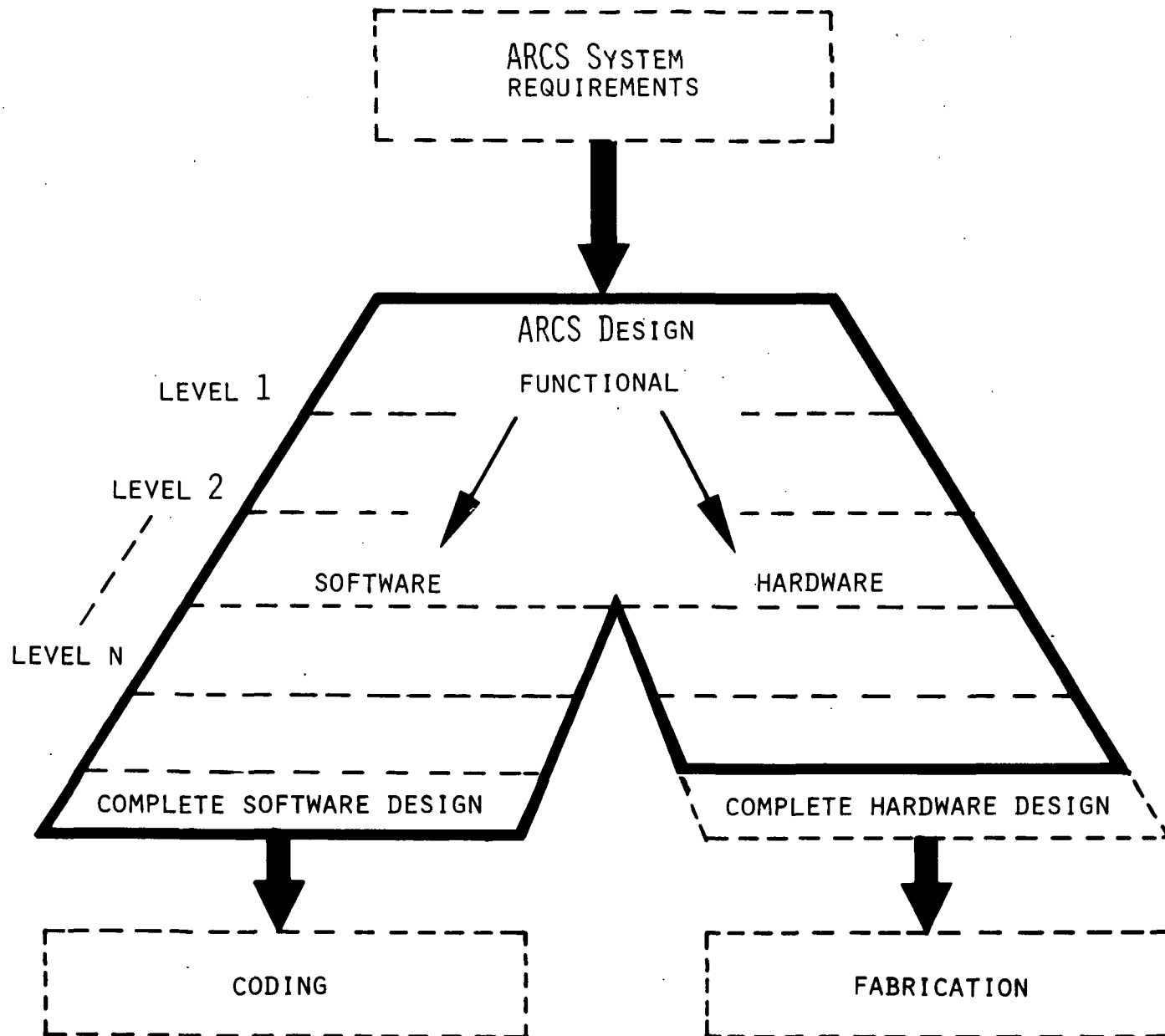


Figure 17.—ARCS Top-Down Design Description

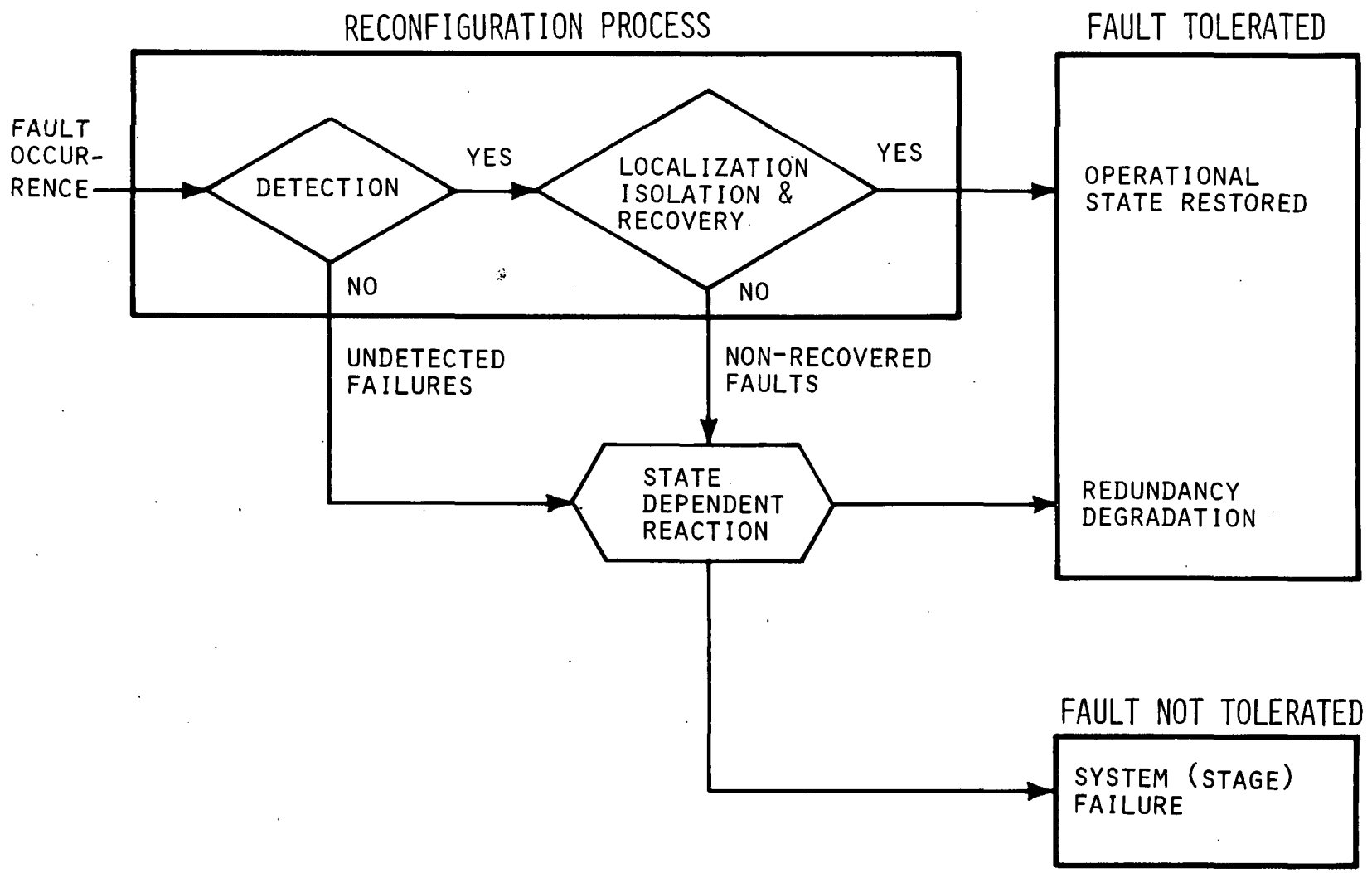


Figure 18.—ARCS Reconfiguration Outcomes

before the fault occurred. Failure of the recovery process, i.e., the fault is declared permanent, will result in a degradation in the redundancy state of the affected stage. A detected, localized, and isolated permanent fault in triplex and duplex, an undetected permanent fault in triplex, or an unrecovered transient fault condition will all result in redundancy degradation. A detected permanent fault in duplex could result in either a redundancy degradation or a system failure depending on whether the fault is localized and isolated. A permanent fault in simplex always means system failure whether it is detected or not.

Crucial to the reconfiguration process is fault detection. Five first-level monitoring functions in the ARCS will immediately, upon activation, initiate action leading to reconfiguration. Three of these first-level monitors—the *computed command output monitor*, the *sensor signal selection/failure detection (SSFD)*, and the *servo monitor*—provide selective fault detection (see fig. 19). The *watchdog monitor* protects against gross faults that render the logic capability of the processor unusable. Whereas these four monitors indicate the occurrence as well as the disappearance of faults, the *power monitor's* primary function is only to indicate that a particular fault condition—loss of power—has ceased, so that recovery can be initiated.

Figure 20 shows in more detail the functional relationships between the first-level monitoring functions and other primary processes involved in the ARCS reconfiguration. The SSFD and output monitor are algorithms implemented entirely in software, whereas the servo monitor is an independent hardware monitor augmented by software functions. The watchdog monitor, which has a primary mission of monitoring the proper execution of software functions, is an entirely autonomous, fail-safe hardware function. All these primary monitors interface directly with reconfiguration process control algorithms—collectively called redundancy management for the purpose of illustration. The SSFD, output monitor, and in part the servo monitor use cross-channel data for comparison in triplex and duplex modes of operation.

In addition to these first-level monitoring functions, hardware monitors, such as a RAM parity error detector and a CPU arithmetic error detector, and a continuously operating self-test function provide second-level monitoring. Information from these monitoring points will be assembled in a fault assessment table. In situations where the first-level monitors indicate a fault but do not provide sufficient information to make a logic decision to initiate reconfiguration, the redundancy management process will refer to the fault assessment table to seek information for fault localization. This will be the standard procedure in duplex operation for an output monitor trip.

Four strategies of reconfiguration are involved in the ARCS. Each is associated with a type of fault or operational circumstance (such as power-on) that forces the reconfiguration process. Power disruption affects the system on a per-channel basis. Faults manifested as sensor signal anomalies are contained by the voting node of the SSFD. Faults within a computer cause output monitor trips per individual output commands, massive output monitor trips, or a watchdog monitor trip. Faults within the servo loop are handled on an individual basis by the associated processor or by the associated independent servo monitor.

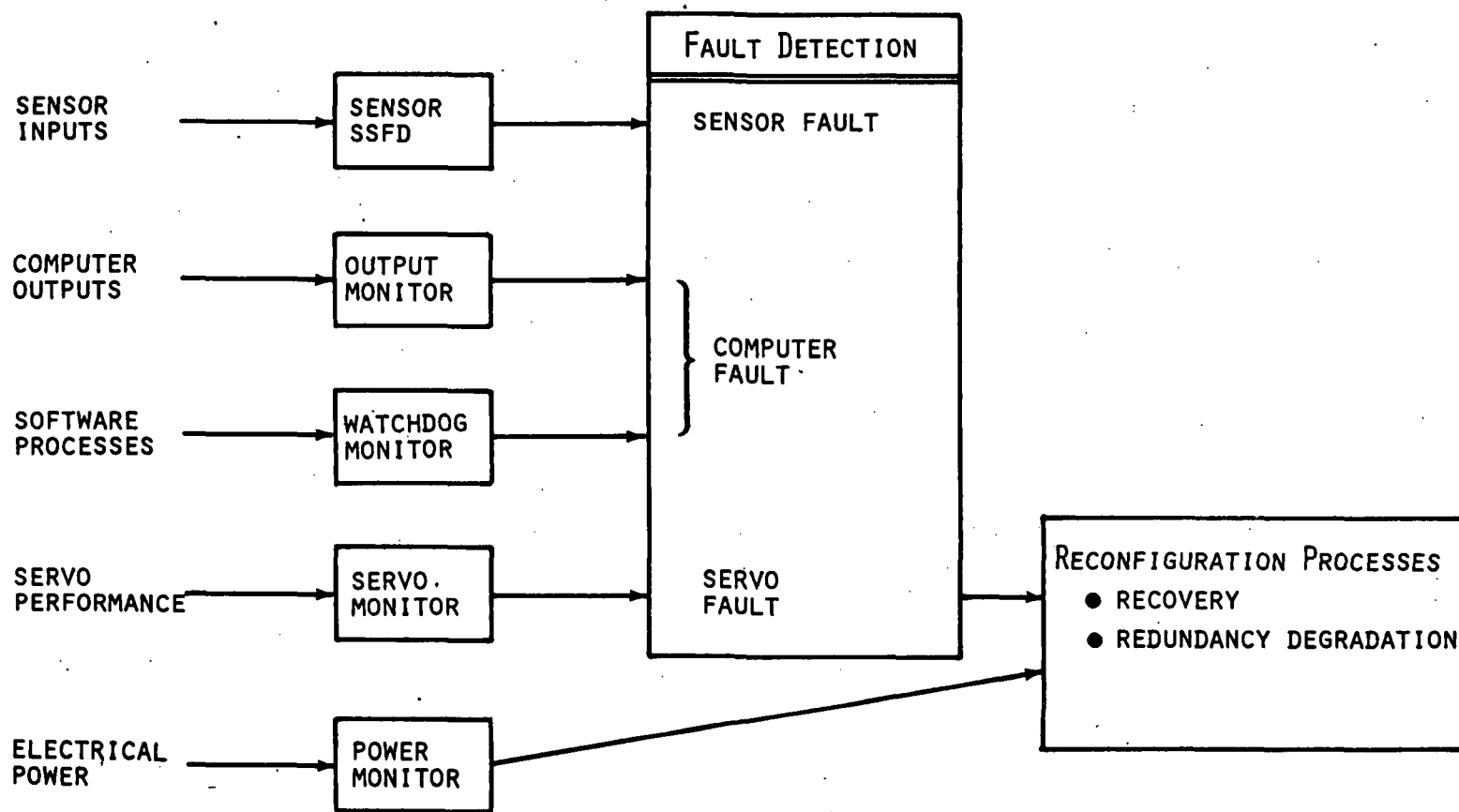


Figure 19.—Paths of Reconfiguration

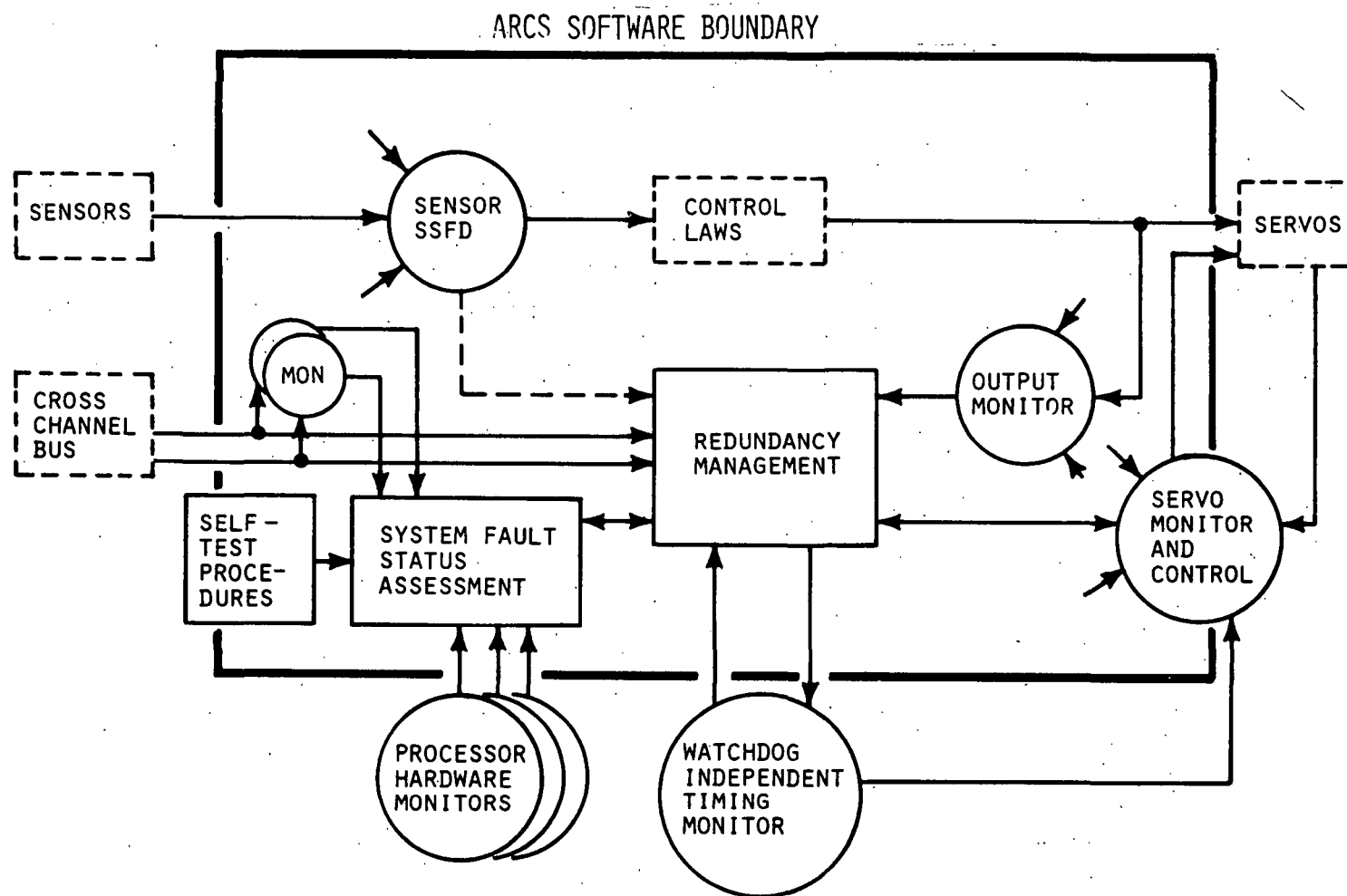


Figure 20.—Reconfiguration Functional Relationships

Two characteristics of the general ARCS configuration governed the choice of strategy for power-on/power-fault recovery. First, the power system of the aircraft is organized on a per-channel basis, with individual circuit breakers for sensors, computers, and servos. The strategy therefore was to accommodate switching on power in any sequence, without any constraints as to intervals between the different switching actions. Second, to minimize the number of line replaceable units and thus cost, the ARCS configuration uses dedicated sensor interfaces per channel with digital, multiplexed cross-strapping of sensor data to facilitate a one-box-per-channel design for the computer. This, however, introduces a sensor dependency on the computer.

The ARCS power-on/power-fault recovery strategy is illustrated in table 1. Each computer's redundancy management keeps a permanent failure flag raised for each of the channels with which it has not achieved frame synchronization. A single operating channel (channel A in table 1) will have B and C permanent failure flags set to mode all its redundancy management processes into single-channel operation. Upon establishing synchronization with a second channel (channel B in table 1), the B permanent failure flag will be removed, the B do-not-use flag will be set, and channel B's processing will be initialized using state variables transferred from channel A.

To prevent the recovery of one computer from interfering with the normal operation of the other computer(s), all state variable data required for recovery is transmitted cross channel every frame. Among the variable data transferred from A to B are do-not-use flags from channel A's SSFD, initially causing both channels to operate on the A sensor data only. As soon as channel B sensor data comes within the SSFD recovery thresholds with A sensor data, the SSFD processing in each channel will mode into duplex operation on a per-sensor basis. Thus the sequence of operation illustrated in table 1. At any time during this process, the logic states in both channels' redundancy management will be identical. Applying power to the third channel will cause the equivalent sequence of events, leading to triplex operation. Loss of synchronization in triplex or duplex will cause the permanent failure flag to be set, thereby causing all redundancy management processes to revert to duplex or simplex modes, respectively.

The output monitor is a software process that compares the computed outputs of a processor with the computed, cross-channel transferred outputs of the other processors. If an output comparison does not agree within the monitoring threshold, an output monitor flag is set for the affected output.

The reconfiguration strategy following an output monitor trip is outlined in figure 21. The same general scheme applies to both the faulted and unfaulted computer(s) in triplex and in duplex. The process shown in figure 21 represents one pass through the program that will be repeated each frame as long as an output monitor disagreement exists. The following describes the process.

Table 1.—Power-On/Power-Fault Recovery Strategy

OPERATION				PERMANENT FAILURE FLAGS		
	SENSORS	COMPUTER	SERVOS	A	B	C
SIMPLEX	A	A	A	0	1	1
DUPLEX AFTER SYNC ACHIEVED	A	A	A	0	0	1
	B	B	B			
DUPLEX AFTER SENSOR SIGNAL RECOVERY	A	A	A	0	0	1
	B	B	B			

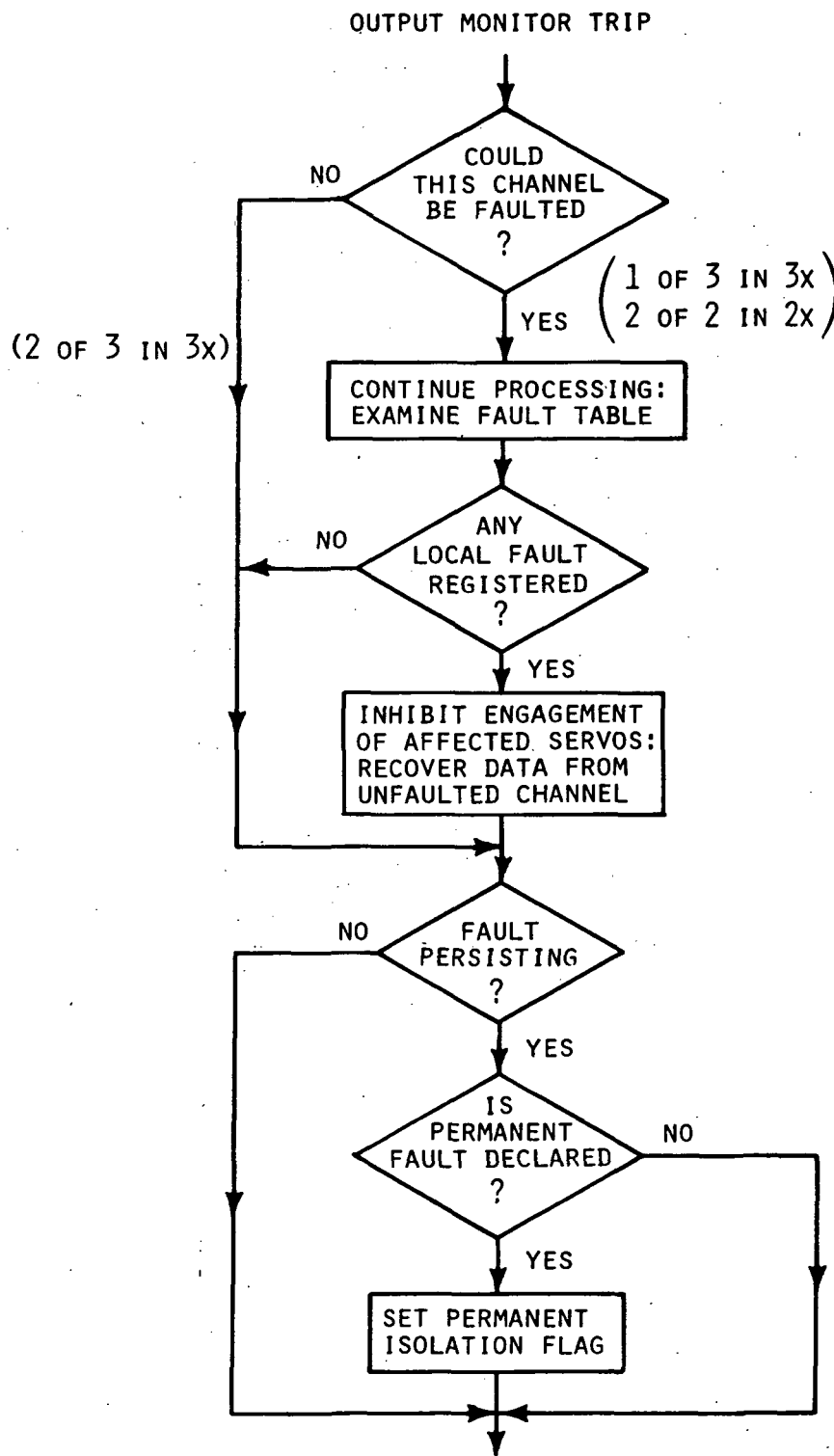


Figure 21.—Strategy Following Output Monitor Trip

First, a computer decides if it could be the cause of the monitor disagreement. The only time it can immediately conclude that it is not the faulted channel is in triplex when a two-out-of-three decision can be made. Concluding that it can be faulted, a processor will check its fault status table for flags that can be associated with the faulting output. If such a fault is registered, the servo management process disengages the affected servo, and recovery is attempted by utilizing variable data from an unfaulted computer to initialize processing during the next iteration.

If the fault persists, as indicated by a continued trip of an output monitor, the described process is repeated for a predetermined number of iterations after which the fault is declared permanent. When this has occurred, a permanent failure flag is raised declaring that the affected output is permanently disabled and the affected servo will be permanently disengaged. If the fault indication disappears before the permanent fault decision occurs, the affected servo will be reengaged.

If a processor's fault status table has not registered a fault that can be related to the output monitor's trip, or if the processor is one of the two unfaulted processors in triplex, the routine to check for a persisting fault and subsequent setting of a permanent isolation flag will still be performed. This strategy provides each operating processor with an independent, permanent failure assessment of the total system.

The result derived from the computer output monitor is an indication of a computer fault and the channel in which the fault occurred. This information is stored in the system failure status table where it can be used by the redundancy management and system test functions. Failure data accumulated in this table is used by system test to record the identity of the failed element for later use by ground maintenance.

The third functional module for which performance monitoring is provided within the ARCS architecture is the servoactuation system. Servo monitoring is provided on two levels—a software monitor of the three coil sum currents per actuator when in a triplex configuration and an independent hardware monitor for each actuation channel. Coil sum currents are proportional to the pressure differential across the actuator and are therefore indicative of the error that exists between that actuator and the force-summed output. By performing a two-out-of-three vote of the three coil sum currents, failure coverage is provided for those failure conditions not otherwise detectable by the independent hardware monitor or the computer output monitor. A failure thus detected will result in a computer-commanded shutdown of the associated actuator.

When a failure condition has been detected by the independent servo monitor, the only indication available to the system software is via the servo engagement discrete. Once a channel disengagement has been effected, the cross-channel monitor of coil sum currents is no longer of use since the bypassed servo will passively track the two good channels. Failure information generated as a result of servo monitoring includes the results of the coil sum current comparison and the detected disengagement status. These pieces of failure data are stored in the system status table for use by the redundancy management and system test functions as indicators of the failure status of the servoactuation system. Since there is a software servo monitor provided for each axis of actuation and an independent hardware monitor for each actuator, fault localization can be effected to the individual actuation channel.

Cross-channel monitoring concepts of the baseline ARCS are based on frame-synchronous operation between channels. The synchronization concept uses a software routine in conjunction with a cross-channel discrete to establish and maintain frame-synchronous processing in all three computers. A synchronization indicator (discrete) is generated by each computer and transmitted to each of the other computers at the beginning of each frame cycle. The sync indicator has the following significance:

- When SET, it informs the other channels that the local channel is initiating the synchronization process.
- When CLEARED, it causes the frame timing reference counter to reset and begin counting out the next iteration period.
- If not SET and CLEARED within a time interval tolerance of the frame real-time period, the local watchdog monitor will trip, indicating an irregularity in the computer's real-time operation.

The local sync routine accumulates failure information about the other computers by testing whether their sync indicators are set and cleared at the appropriate times. This information is passed to the local redundancy management process, which will use the sync information, as well as data from other sources, to arrive at the overall system failure status. The sync routine is processed once every minor frame, assessing the other channels' operations each frame regardless of their previous failure status.

The watchdog monitor is an independent, fail-safe monitor of the real-time operation of each computer. If the measured time interval between the clearing of consecutive sync indicators is not within specified upper and lower limits of the nominal frame time period, the watchdog monitor will indicate a computer fault condition. If the computer's sync indications return to a periodic interval, the watchdog monitor will clear the fault indication.

A cleared state of the local watchdog monitor is required for the independent servo engage logic to respond to engage command discretes from the computer. When, as a result of a fault, the watchdog monitor trips, all servos in the local channel will be disengaged. If the fault clears and the watchdog monitor is subsequently reset, the computer must reengage all the local servos.

The relationship between the minor frame iteration timing reference, the sync indicator setting and clearing, and the watchdog monitor function is shown in figure 22. A timer interrupt occurs when the frame iteration timing reference reaches the end of a frame time count initiating the sync routine. The local sync indicator is subsequently set and, while the local indicator is high, the local channel is testing the other channel's sync indicators to determine if they are ready to sync. When all indicators are set, all three channels clear their local sync indicators essentially simultaneously. This resets the iteration timing reference in each channel and all three computers are thus synchronized. The processors then independently execute their in-line program until the timing reference again reaches the end of the minor frame time count.

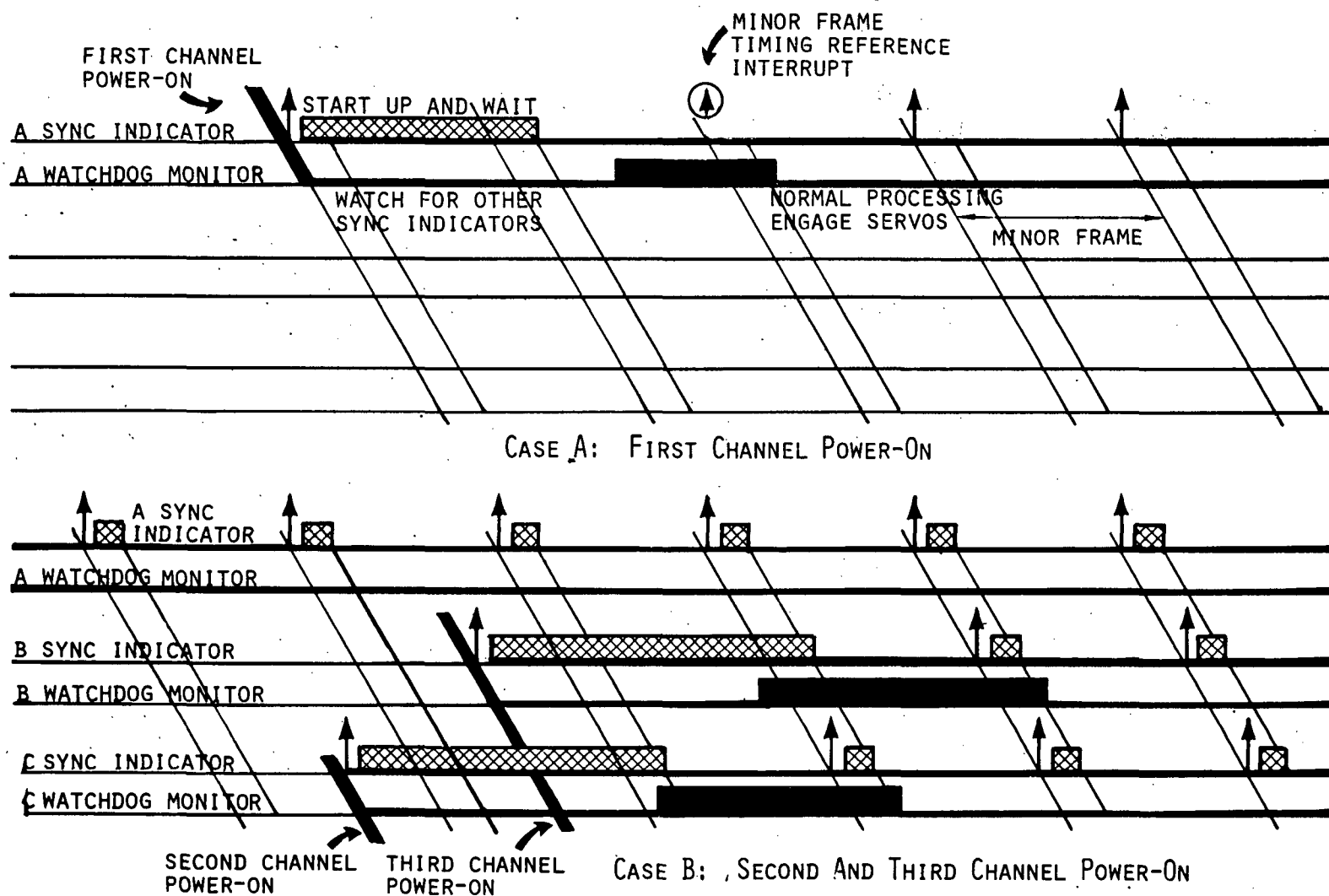


Figure 22.—Relationship Between Iteration Timing Reference, Sync Indicator, and Watchdog Monitor

Should the processor fail to set or clear its sync indicator, the watchdog monitor will trip and the minor frame iteration timing reference will not be reset. The iteration timing circuit continues to count and will cause a local interrupt every time it overflows. If the computer is able to respond to these interrupts, they will be interpreted as recovery interrupts by virtue of the fact that the watchdog monitor has tripped. The redundancy management process will then attempt a recovery operation.

In the ARCS recovery process there is no difference between resynchronization initiated by the redundancy management function and initial sync as a result of a power-on interrupt. If the local channel determines that another computer is operating normally, it waits for the beginning of the next frame's computation, as signaled by a foreign sync indicator going from clear to set, and enters the normal sync routine. It then enters the operational software at the appropriate minor frame as part of the recovery operation.

During an initial startup, or during a recovery attempt, the local channel must first determine if any other computers are operating. It does so by monitoring the sync indicators of the other two channels for a period of time longer than one frame, as illustrated in figure 22. If the local channel detects no activity by the other computers, it goes into a simplex mode of operation. The initial sync process proceeds through the following steps:

- Mask timer interrupt.
- Set local sync indicator.
- Test if other sync indicators are set.
- Wait for 1.1 frame times while testing other sync indicators to determine if there is any activity.
- Clear local sync indicator.
- If local computer determines that it is the only one operating, it should:
 - Enable timer interrupts.
 - Start processing.

Otherwise it should:

- Wait and sync with an already operating channel.
- Enable timer interrupts.
- Copy state variable data.
- Start processing.

The normal synchronization process repeated in each frame includes the following steps:

- Check sync indicators for clear upon entry to routine.
- Set local sync indicator.

- Wait, alternately testing the two other sync indicators, until one of the other computers is ready.
- If neither is ready before the time limit is exceeded, interpret this as loss of sync.
- After one of the other channels is detected, wait for last channel.
- If the time limit is exceeded for the third channel, mark it failed and continue.
- Clear local sync indicator.
- Test sync indicators to be sure that none failed in a set condition.

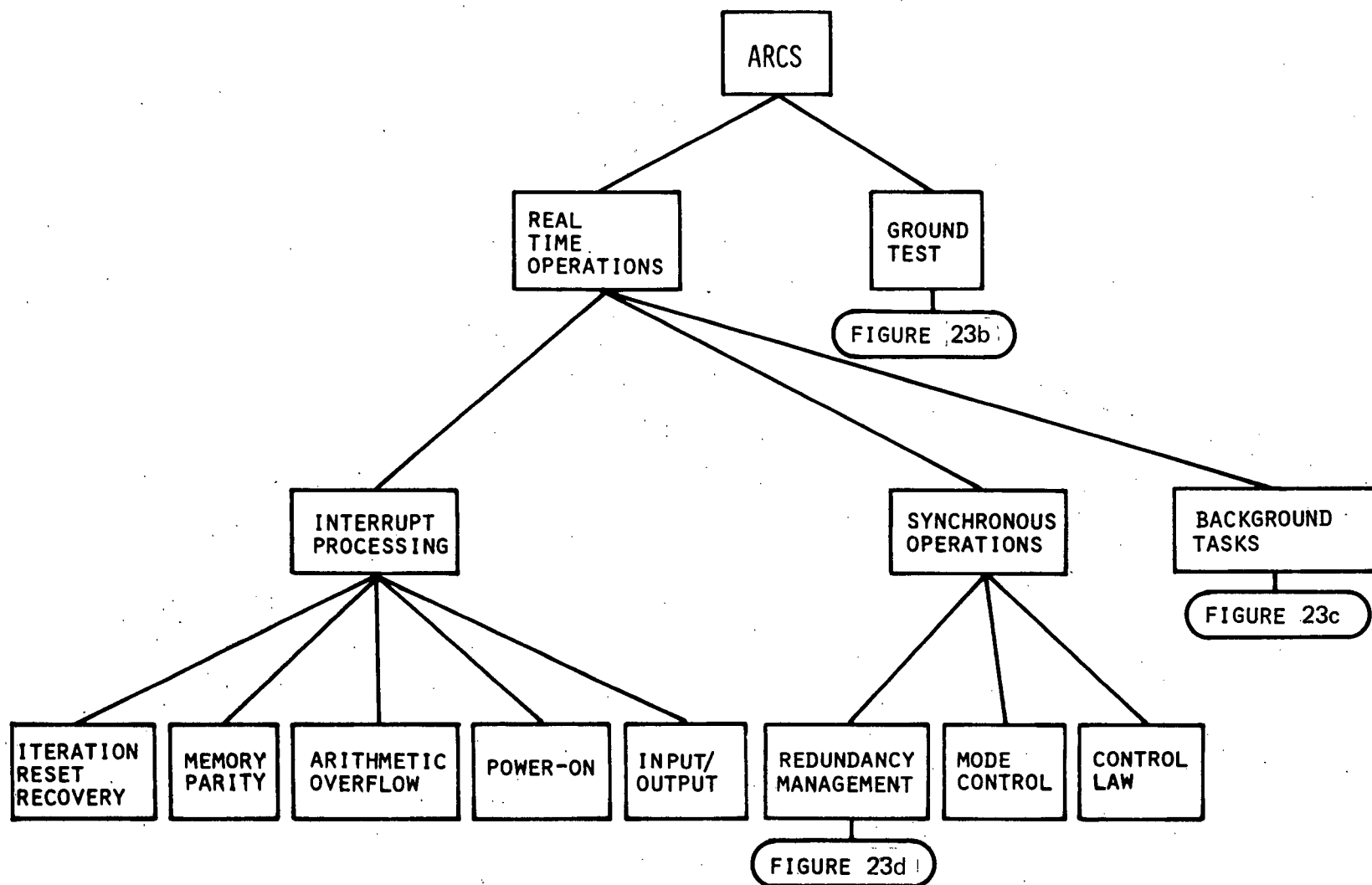
Execution of the synchronization function in software simplifies the hardware by eliminating the necessity of interconnecting the three iteration timing references and voting on them in each channel.

In this section we have identified primary elements of the ARCS redundancy management processes and defined some of them on a functional level. In the following section we will organize all the processes to be performed by the ARCS in a systematic manner so that the visibility relative to requirements is maintained down through all levels of design.

5.1.2 FUNCTIONAL ORGANIZATION

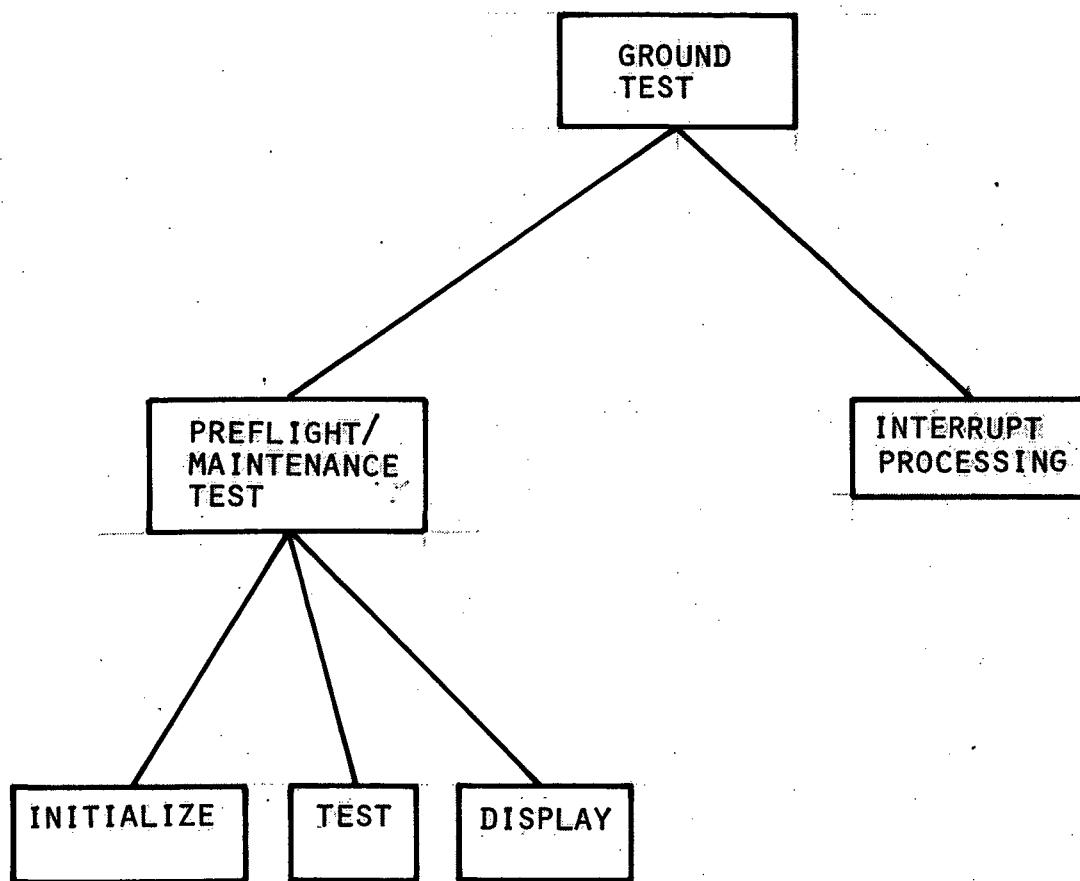
We have chosen the format of a tree, as shown in figure 23 to subdivide the overall ARCS process into progressively smaller processes that can ultimately be implemented as a hardware or software algorithm.

The first level of breakdown separates real-time and non-real-time operations within the ARCS (fig. 23a). Real-time operations imply that some constraint relative to the time available for function execution is a primary consideration in the requirements set. Real-time tasks are all those ARCS processes that take place during the operation of the airplane. The only non-real-time ARCS tasks are ground test operations for preflight verification of system integrity and maintenance (fault identification and verification of maintenance actions). Figure 23b illustrates the functional breakdown of the ground test operation.



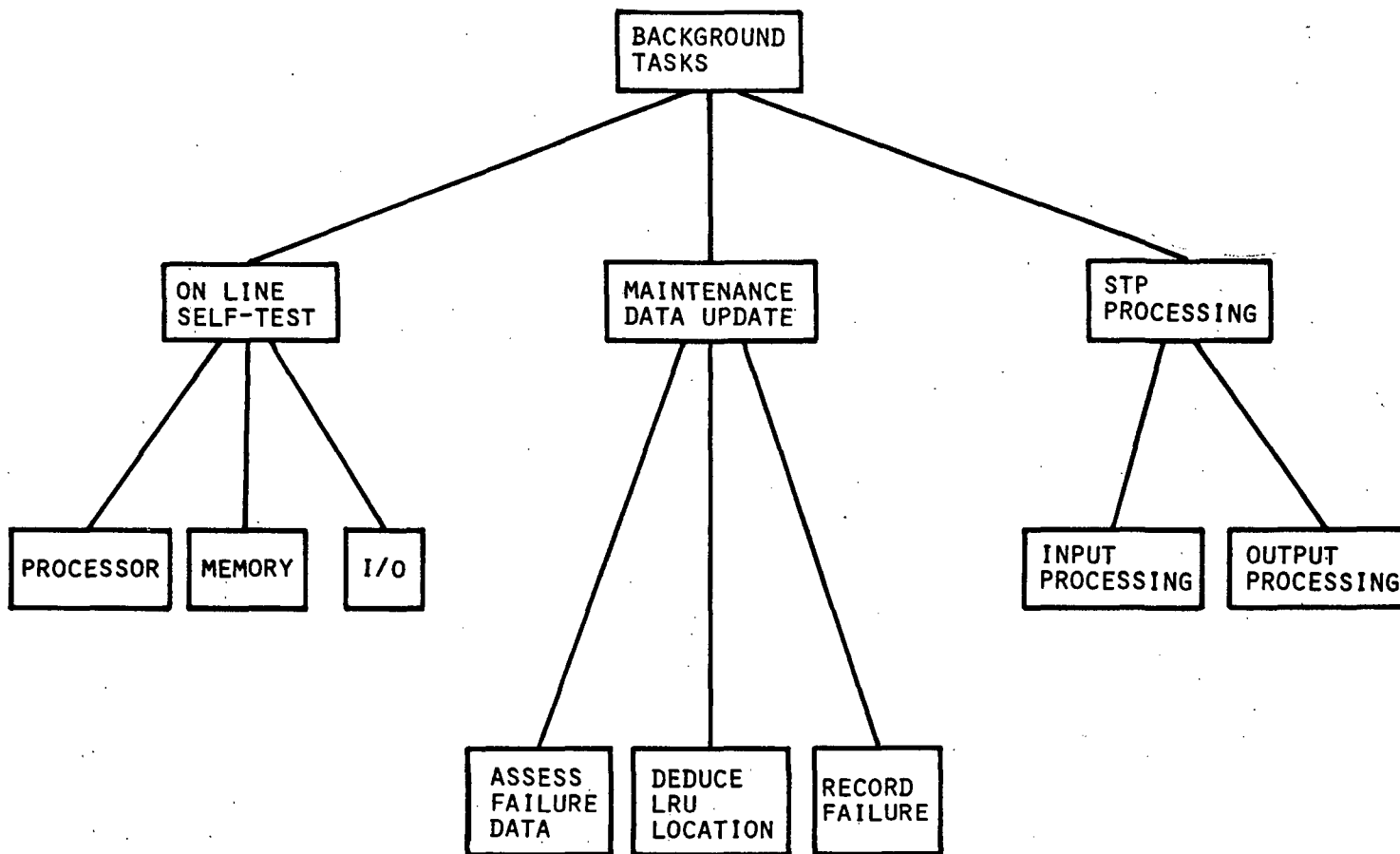
(a) Overall ARCS Process

Figure 23.—ARCS Functional Tree



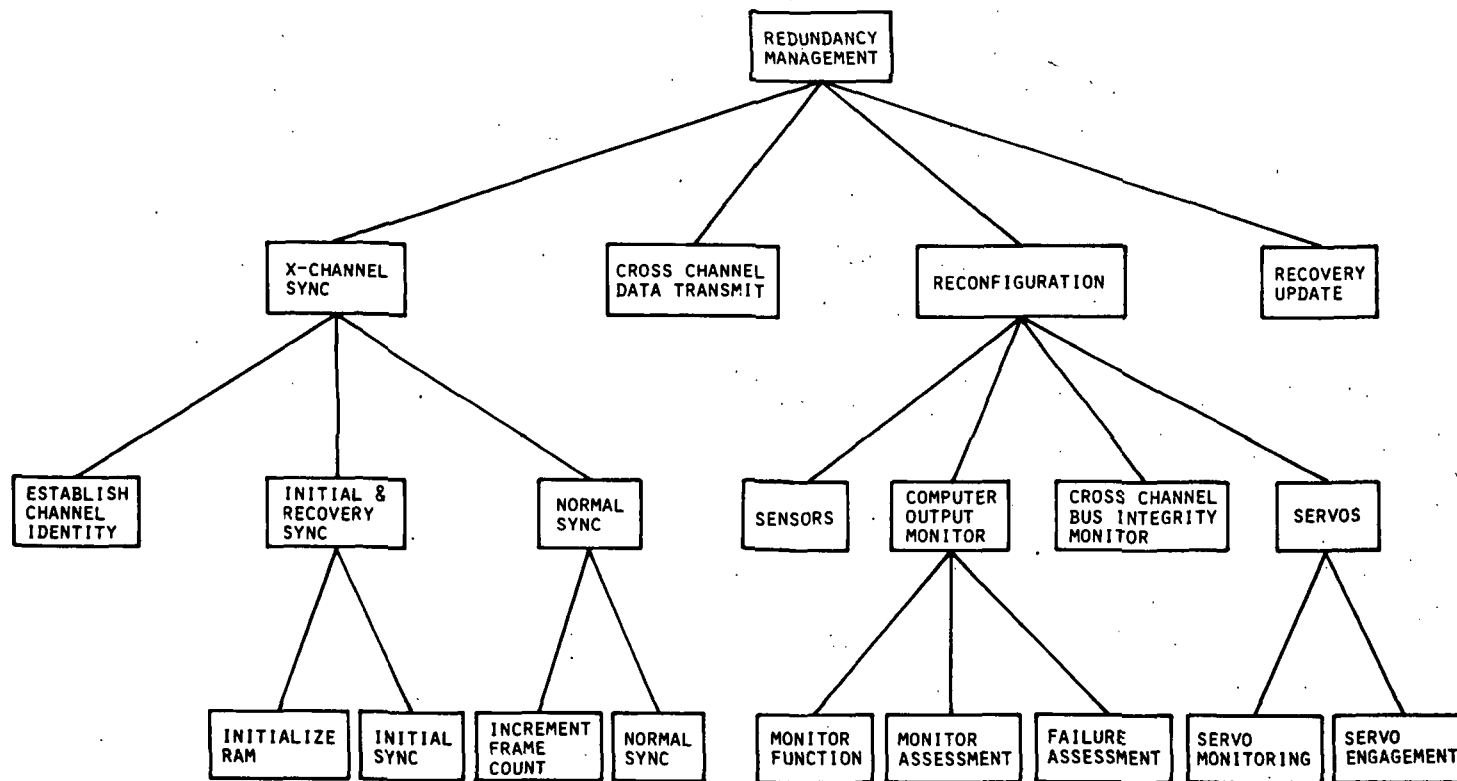
(b) Ground Test

Figure 23.—Continued



(c) Background Tasks

Figure 23.—Continued



(d) ARCS Redundancy Management

Figure 23.—Concluded

Preflight testing is conducted by the flightcrew as a part of the routine predispatch airplane checkout. The purpose is to assess whether dispatch-required functions are operational. The results of such a test will be a "go" or "no-go." The purpose of postflight or maintenance testing is threefold: to test for and register the existence of hard failures that exist within the system, to test for latent failure conditions that cannot be checked during on-line operations, and to verify system operational integrity following maintenance action. This testing is provided for use by the ground maintenance operations, and hence the failure data thus generated will be aimed at identifying the specific LRU in which a failure has been detected.

A transition from real-time operations to ground test will occur only when such a request has been generated by the operator and the aircraft is stationary on the ground.

Since ground testing is essentially a non-real-time operation for the majority of the test sequence, no attempt is made to maintain normal synchronization between the various processors. Furthermore, computer subsystem testing involves RAM memory write/read testing, which means that any state variable data previously stored in memory will be lost. A computer recovery process is consequently initiated upon return of control to real-time operations.

The second-level breakdown of the real-time operations reflects the digital implementation of the ARCS and the necessity for processing some of the functions within the redundant channels with a certain degree of time synchronism.

Synchronous tasks include all those functions that have to be performed in a parallel manner in all channels to achieve efficient cross-channel consolidation at regular time intervals. These functions are also referred to as foreground tasks. Asynchronous tasks, or background tasks, are those tasks that, within certain limitations, can be performed on a time-available basis. Interrupt processing tasks are those software functions triggered by events external to the software. These events include iteration timing reference timeouts, I/O interrupts, hardware faults, and power-on monitor trips.

Asynchronous real-time ARCS tasks are those monitoring functions, performed entirely within each channel, that are necessary for establishing system integrity during aircraft operation. In-flight (or on-line) testing of the computer subsystem is the primary function performed by the asynchronous task. Its purpose is to detect and/or localize those failure conditions that might not be detected by first-level system monitors. In-line monitoring is provided both to enhance fault tolerance and to augment fault localization for maintenance and repair. To achieve high second-failure coverage, a rapid means of localizing faults in duplex operation is necessary. The in-flight test is a continuous, repetitive check of the computer subsystem that keeps a running record of anomalies within the local computer available as additional information to the redundancy management algorithms in resolving conflicts in duplex configurations.

The design of the ARCS asynchronous operations follows the design requirements tree shown in figure 23c. The three major asynchronous functions are on-line self-test, maintenance data update, and system test panel (STP) processing. Because of the obvious benefit to system maintenance of recording functional anomalies occurring during operation, asynchronous tasks include algorithms to identify and store information for later use in establishing required

maintenance actions (maintenance data update). Algorithms necessary to provide interface between the crew and the system through the STP (STP processing) also fall in the category of asynchronous operations. A more complete description of the self-testing aspect of the asynchronous task is contained in section 5.1.4.

The core of the fault-tolerant system, redundancy management, is contained in the synchronous operations, which also include the application tasks of control law and mode control processing shown previously in figure 23a. The projected application tasks for the ARCS, identified in section 4, consist of those control computations associated with flight-crucial and flight-critical control functions:

- Flutter suppression
- Structural mode suppression
- Maneuver load alleviation
- Fly-by-wire control
- Stability augmentation
- All-weather autoland

Redundancy management, shown in figure 23d, breaks down into the four processes of synchronizing the redundant channels, exchanging variable data between channels, substituting data from an unfaulted processor during recovery attempt, and revising the redundancy configuration in response to power-on/power-off states and fault occurrences.

Reconfiguration processes to tolerate transient and permanent faults involve three major stages of the system related to the location of the fault. Reconfiguration due to faults located functionally upstream of the software sensor selection voting node are handled by the SSFD. Faults in the processor will manifest themselves at the output monitor or watchdog monitor, and the appropriate reconfiguration process, described earlier, will be performed. A special class of faults, namely those associated with a corrupted cross-channel data bus, will cause output monitor trips despite correctly operating processors at each end. A cross-channel bus integrity monitor provides failure localization information to the reconfiguration process to handle this class of faults.

The third major class of faults includes those located in the servos. The reconfiguration process for servos involves the monitoring and engagement/disengagement of the individual servo loops.

In this and the previous section, we have introduced the general system reconfiguration strategies and the overall ARCS functional organization. Before progressing with the detailed software design description, we will develop in more detail the functional principles of two processes within the reconfiguration domain; sensor signal selection and failure detection (SSFD) and system self-test.

5.1.3 SENSOR SIGNAL SELECTION AND FAULT DETECTION (SSFD)

As implied by the name, the SSFD algorithm has two primary functional objectives: (1) to extract the most useful data from the redundant set of signals and (2) to determine if any input signal is operating outside normal tolerance limits and, if so, isolate it so it does not influence the output from the algorithm.

The ARCS processors operate in frame-time synchronism. The signal selections performed on the continuous and discrete sensor-inputs provide a consolidation point for the redundant sensor data so that all processors operate on identical data and therefore perform identical processes with identical results. Thus a simple fault-detection scenario is provided for the stage downstream of the SSFD output, i.e., any discrepancy between channels indicates a fault condition.

Sensors are not perfect, however. Each sensor type is afflicted with its particular set of error characteristics: bias error, scale factor tolerances, dynamic response tolerance, and noise. Such errors, if not compensated for or eliminated, must be tolerated by the failure monitoring. This means that fault detection thresholds and detection delays must be large enough to accommodate legitimate differences between the redundant signals, and the fault detection capability becomes compromised.

To enhance fault detection, the ARCS SSFD algorithm therefore incorporates bias error compensation for all sensor signals and scale factor compensation for certain sensor signals where scale factor tolerances may have a significant effect on fault detection thresholds. An example in this category would be bank angle, which normally operates near zero but occasionally reaches up to 30° during maneuvering.

The reconfiguration function of the SSFD algorithm performs the same processes as those described for the ARCS as a whole in section 5.1.1: fault monitoring and detection, fault localization, temporary isolation, and permanent isolation of the fault. Fault monitoring is based on comparison between the redundant signals for first and second fault. Channel localization of first fault in triplex is automatically done through fault detection, but determination of fault location in duplex requires information derived from within each channel. This determination uses sensor valids if available, reasonableness tests, or comparison with signals synthesized from state vector data available within the processor. Faulty signals are isolated by modifying the algorithm to ignore the faulty input.

The continuous-signal SSFD algorithm incorporating bias error compensation defined for the ARCS is shown in block diagram form in figure 24. Each processor performs the processing depicted for each type of sensor. The total algorithm consists of four subprocesses, three of which are repeated for each of the three input signals.

First, the bias error calculated during the previous iteration is subtracted from the raw signal to provide a compensated signal. The difference between the compensated signal and the average from the previous iteration is then used to monitor for dynamic faults, i.e., a rapidly deviating raw signal input. In order to tolerate legitimate dynamic differences under certain operating conditions, dynamic fault detection may be suppressed by delaying the monitor

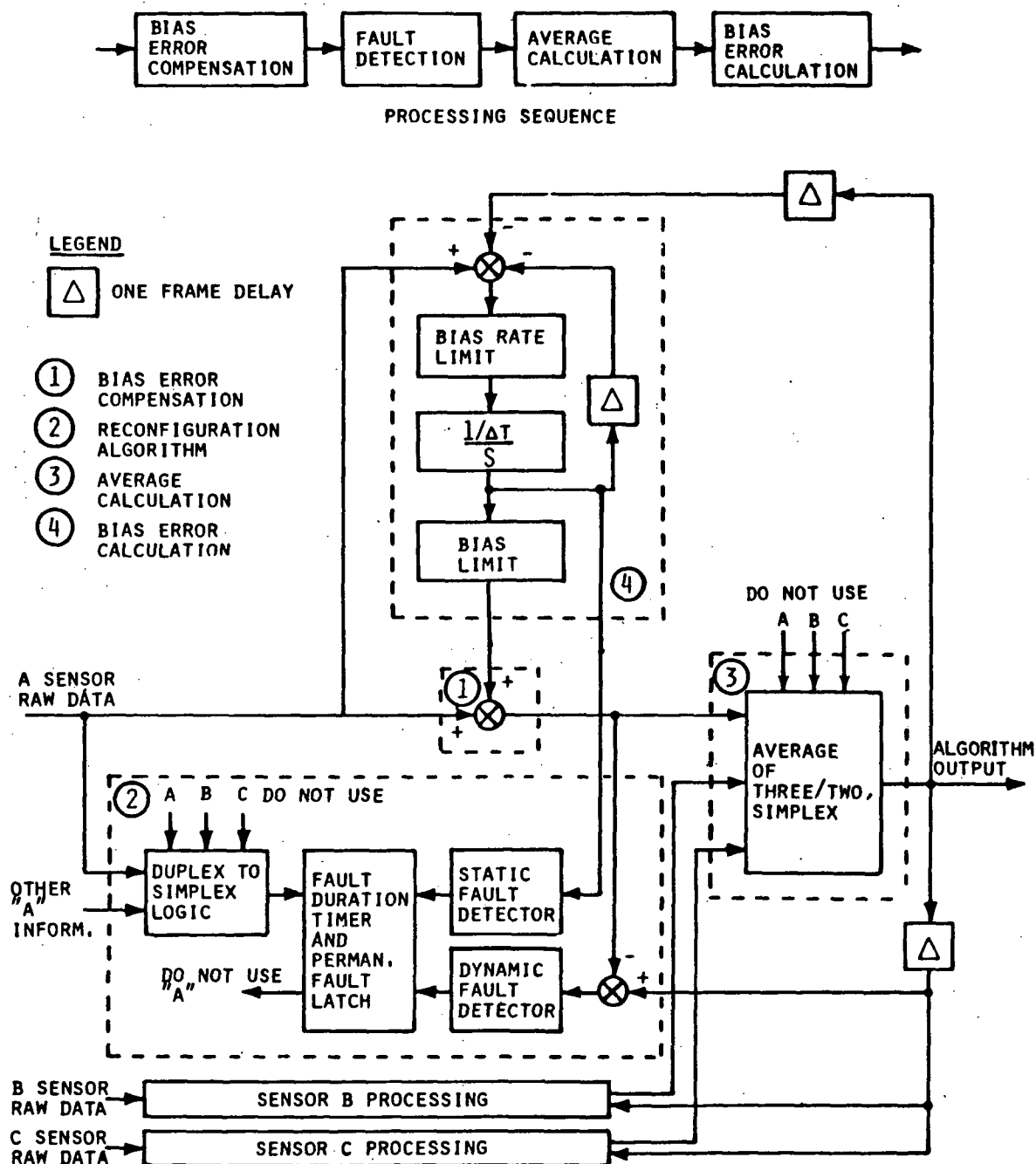


Figure 24.—Continuous Signal Selection/Fault Detection Algorithm

trip for some sensors. The dynamic fault detection is part of the SSFD reconfiguration algorithm. The output of the reconfiguration algorithm is a do-not-use flag against the raw signal input.

After monitoring is completed for dynamic faults, the three (or two remaining after first fault) compensated signals are averaged to derive the SSFD algorithm output.

The difference between the new average and the individual raw signal is input to the bias error calculation to be used during the next iteration. The bias error compensation is primarily a low-pass filter with the purpose of letting through only the static value of the signal difference. A bias rate limit and a bias magnitude limit are included to protect against a signal buildup should the raw signal fail actively. The bias error is used to monitor for static fault. When the bias error reaches a predetermined level, fault is declared.

A trip of the static or dynamic fault detector is an input to the SSFD reconfiguration logic. The first step will be to raise the do-not-use flag for the faulty signal. Even if the signal returns below detection levels, this flag will be kept raised for a predetermined time period T_1 , starting from the moment the signal returns, to prevent oscillatory failures from passing through the selection process.

Should a second fault occur while the first do-not-use flag is raised, the first fault will immediately be latched as a permanent fault.

The duplex-to-simplex logic will be configured according to other information available for the particular sensor. For sensors with high-confidence internal monitoring, the valid signal may be a sufficient data item on which to assume whether or not the sensor has failed. For some sensor signals, functionally redundant information for comparison can be synthesized from other sensors in the system or, more generally, from the system state vector.

The discrete SSFD algorithm, shown in figure 25, is simpler since it operates on only two-valued signals. Time skew between signals is the major complication, in providing a non-ambiguous output from the algorithm.

Each of the inputs is compared with the algorithm output to determine if change of state is required. The comparison output is majority voted, with the result delayed to suppress transient anomalies caused by contact bounce and noise, before a change of state is executed.

Fault monitoring is performed on each of the input/output comparisons. Any persistent disagreement indicates a fault and will cause the associated do-not-use flag to be raised.

Information derived by the SSFD that is pertinent to the maintenance operation is the identification of sensors that have been declared as permanently failed. The failure information thus generated is used by the system test function to provide LRU-level failure identification that will be stored in a nonvolatile section of memory for later use by line maintenance.

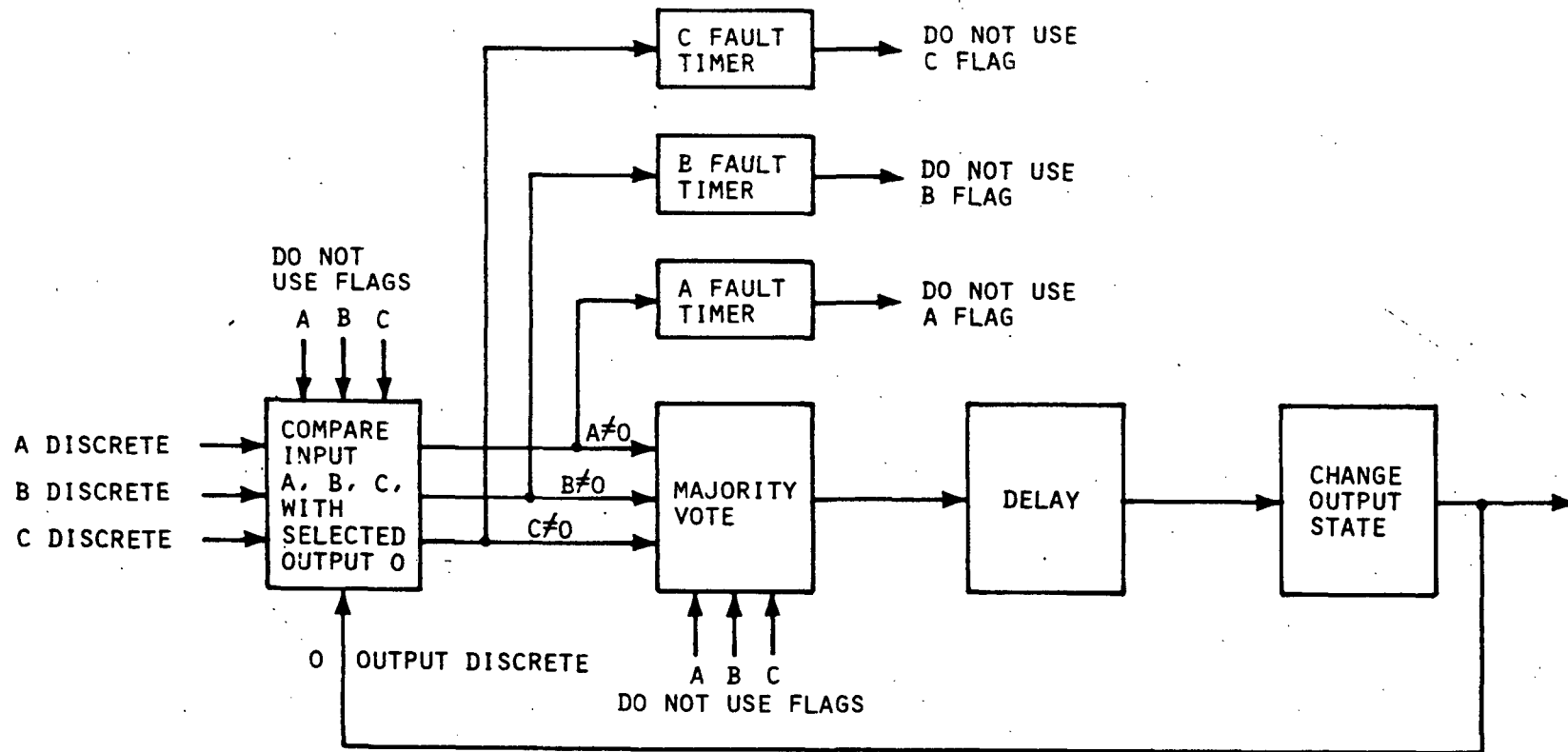


Figure 25.—Discrete Signal Selection/Failure Detection Algorithm

5.1.4 SYSTEM SELF-TEST

Second-level monitoring, performed on-line during in-flight operation to provide fault information for the redundancy management function to augment fault localization after fault detection by a first-level monitor, consists of software routines and hardware monitors. This monitoring is concerned with the integrity of the processor, memory, and input/output interfaces.

Processor self-test is the primary on-line software self-test for the computer unit. Although it runs in the background mode, it must be completed repetitively as determined by the time deadline, i.e., a maximum time is specified between sequential completions of the test. The computer self-test begins with an instruction test sequence within which all instructions are exercised, all registers are involved, and all addressing modes are used.

Endless loop instructions are included at points in the program where an improper computation might mislead the processor. If the processor performs an erroneous computation and goes into one of these endless loops, the time deadline for the computer self-test will not be met.

The second test to be performed in the background mode is a program memory sum check, wherein all the words in the program memory are summed and compared with the known correct sum. This test is provided for verification of the overall integrity of the program or constant memory, but it could conceivably be partitioned by control axis or mode for improved fault localization if such resolution is possible with the program architecture.

The third test in the computer self-test sequence is a scratch-pad read-write test. A number of locations in the scratch pad are dedicated to self-testing. On successive iterations of the test, random patterns are written into these dedicated locations and then checked. This test is designed to test memory integrity and addressing structure throughout the scratch pad.

Proper operation of the computer input and output sections will be tested using wraparound loop checks of both analog and discrete data. Analog I/O will be verified by comparing a special test output with the same output data that has been looped back into the computer through the multiplexed A/D. Because each of the servo output commands are expected to remain approximately equal to zero, massive "stuck-at" faults of the multiplexed portions of the analog I/O will be detected by varying the value of the special test value with each pass. Operation of the discrete I/O will be similarly checked by one dedicated test discrete output that will be looped into a dedicated input channel. This discrete value will alternate between a 1 and 0 state, again to detect "stuck-at" faults of the multiplexed elements of the discrete I/O.

Each processor can operate a loop test through the interchannel data link by transmitting a sequence of known random bit patterns to the other processors as a part of the normal cross-channel data transmission. The other processors must return the pattern unchanged to the originating processor. This test is used to provide integrity checking of the interchannel data links.

When a failure is detected during the in-flight test, the failure condition will be recorded in a temporary storage register. When the failure is recorded, control is returned to the test program to continue testing. In this way the entire test sequence can be completed and all failure information accumulated before any update is made to the maintenance data.

All failure data generated by the on-line self-testing function is stored in the system status table as well as the maintenance data table. This information is then available to the redundancy management processes for fault localization purposes.

A further aspect of system self-test is the ground test operation whose purpose is to verify total system operational integrity for preflight verification or following system maintenance. Any system failure conditions detected during ground test will be analyzed and the data formatted for display on the STP, with the particular display depending on the failure condition and the test mode selected.

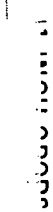
The procedure of ground testing, whether it be for preflight or maintenance purposes, makes use of a so-called "center out" test philosophy, wherein the most basic elements are tested first and then these basic elements are used to test other functions. This is illustrated by the ground test functional requirements free in figure 26. At the highest level, this test structuring results in an organization that tests the computer LRU first, then the input/output electronics sensors, and finally the servo systems. The computer testing segment is further broken down into its constituent parts, i.e., central processor element, computer hardware monitors, and RAM and ROM memories.

Following a diagnostic test of the processor as previously described for on-line self-testing, the ability of each of the hardware monitors to detect and annunciate the existence of a fault condition is verified. The ability of the watchdog monitor to indicate a "good" state and a failed state will first be verified. Next, the ability of the arithmetic error detector to detect an overflow condition and generate the corresponding interrupt will be verified.

A parity generator inversion discrete, controllable from within the processor, is provided in the ARCS. Setting this discrete will cause the words written into memory to be stored with even parity. When an attempt is made to read this data out of memory, a parity error interrupt should be generated. Where separate parity error detectors are provided for different sections of memory, this test will be repeated for each parity detector in the system.

The ROM or constant memory will be tested using sum checking techniques as described for on-line self-testing. The operational integrity of the write and read aspects of RAM memory will be verified by use of a predetermined data pattern. By comparing the pattern readout with the pattern that should have been written in, failures within the RAM memory can be detected. In addition to comparing data patterns, every write and every read cycle passes the data through the hardware parity generator and parity checker, respectively.

In the next step, a test will be conducted to verify the operational integrity of the computer's output and input electronics. The ARCS I/O can be roughly divided into three categories: analog, discrete, and digital. The approach used is what is commonly referred to as wrap-



1

around testing, whereby a specific output is commanded by the processor and the resultant output looped back into the corresponding input processor. The received data can then be compared with the transmitted data to check both input and output functions.

For analog data, a single wraparound test path is sufficient to verify all the common or multiplexed elements of the input and output electronics. However, such a test does not check a significant amount of electronics downstream of the demultiplexer on the output (namely, individual sample and holds) and upstream of the multiplexed A/D (such as individual buffers, demodulators, and filters). These elements are typically dedicated to specific inputs and outputs. The proposed baseline ARCS hardware includes the capability to switch, under software control, all the analog output commands into all the analog A/D input channels. In this way all I/O channels can be tested over their full range as a part of the ground test routines.

A similar capability has been provided for discrete data by switching all discrete outputs into all discrete input channels; digital I/O data uses a hardware loop from the cross-channel data transmitter to receivers. All such loop testing is accomplished under software control.

In addition to the intrachannel digital I/O loop check, each processor can operate an interchannel loop test through the interchannel data link. A sequence of known bit patterns is transmitted to the other processors, which must return the pattern unchanged to the originating processor. This test serves two purposes: to provide an integrity check of the interchannel data link and to assess the operational redundancy level of the computer subsystem.

To verify sensor system functional capability requires that at least its output be stimulated to produce some expected value. To do this without using ground support equipment (GSE) implies that, wherever possible, a sensor's own internal self-test be used to provide this output. Based on the expressed desirability of simplicity, speed, and one-man operation, automatic stimulation of sensor self-test functions will be provided wherever possible. Where automatic stimulation of self-test features is not possible, the interactive approach, wherein the test operator is requested via the STP to provide the required stimulus, is used to maximize test effectiveness.

The servoactuator is the third major functional block to be checked out during ground test. A functional breakdown of those tasks required to check out the operation of any given channel of servoactuation includes verification of the engagement/disengagement function, dynamic response testing, and verification of the force voting feature. The testing tasks/procedures must be repeated for every axis of actuation.

5.2 SOFTWARE DESIGN

The functional concepts described in the previous section were translated into a software design using the methodology described below. The ARCS software design is further illustrated in appendix B down to a level relevant for the understanding of the fault-tolerant operation of the baseline ARCS, i.e., the application software is described only to a level required for understanding its overall place in the complete design.

Reliable software is an obvious requirement for a fault-tolerant computer system. Reliable software requires a strict software design methodology. The software design methodology adopted for the ARCS design process recognizes that the single most important factor in achieving reliable software is to maintain design visibility on all levels throughout the design. This visibility is necessary to ensure that all requirements are being correctly interpreted and met, as well as to provide continuity across interfaces between personnel involved in the development, testing, certification, and continuous operational maintenance of the system.

The software design process has been set up as a sequence of steps. In practice, each step may be repeated several times. Each step includes appropriate documentation. The steps are as follows.

1. Perform the top-down functional identification of the system design and generate the associated process function tree.
2. Draw the software structure definition tree.
3. Enumerate all modules of this tree and compile the data-space elements for each module.
4. Starting with the module for level 1, construct the intramodule transition diagram for each module of the system.
5. Generate the software code from the transition diagram in a top-down manner.

Details of these design steps are presented below.

5.2.1 TOP-DOWN DESIGN (STEP 1)

Top-down design is the process of systematically defining processes that satisfy the requirements of the desired system. The output of this step is a description of each process that was defined and the process function tree. This tree is simply a drawing that shows the top-down relationship of the functions, and it will be used in step 2 to generate the software structure definition tree. A typical process function tree is shown in figure 27.

5.2.2 SOFTWARE DESIGN TREE (STEP 2)

The requirements tree of step 1 is used to generate the software structure definition tree. Identical to the function tree, this tree is used to provide short names for the functions and to define the modules of the system. A typical tree and its modules are shown in figure 28.

A module is a collection of related processes. The processes in a module share a common data space that is defined for the module. Figure 29 shows a module and the relationship with its submodules. A module is typically only a link to its submodules and as such does not perform any computation. The lowest level processes are the ones that actually do the computations.

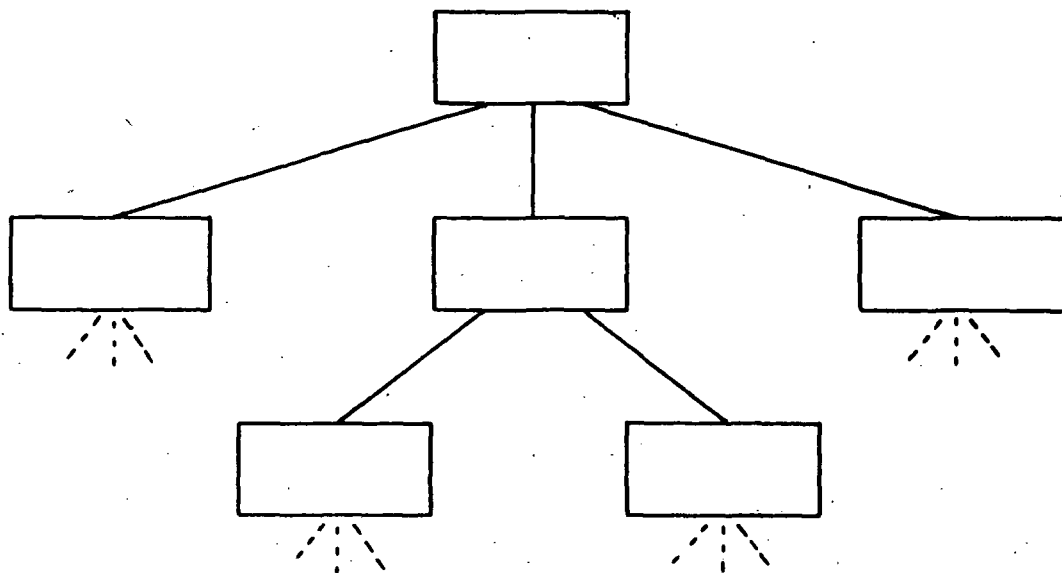


Figure 27.—Process Function Tree

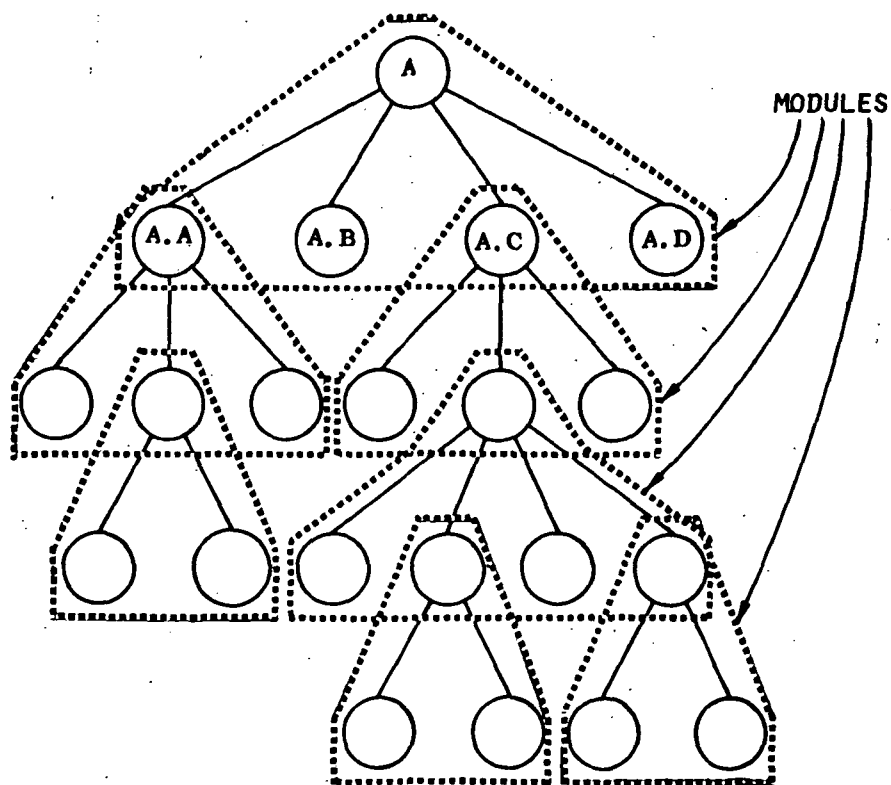


Figure 28.—Typical Tree and Its Modules

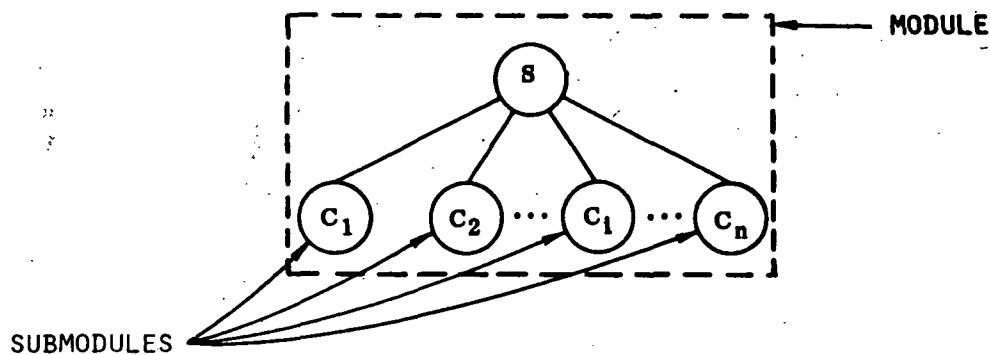


Figure 29.—Software Module

In step 4, a transition diagram for each module will be constructed. The transition diagram shows intramodule control transfer. Control transfer from a module to its submodule is to the single submodule with a transition in the transition diagram that does not have an origin state. The transition back to the supramodule is defined by the module in the transition diagram that has a transition with no destination state.

Note that in such a tree as shown in figure 28, a module will take the structured name of the supramodule. This notation will facilitate discussing the design and relating the design to the software.

5.2.3 MODULE IDENTIFICATION (STEP 3)

In this step a table of data-space elements is defined for each module. To aid in defining these data-space elements, one should consider the input and output requirements of the submodules. These elements are the data to describe the state of the system, tables to record features of the execution, arrays of data, etc. The entries in the table have the form <short name>, which is a description of the data item such as dimension of the array, the meaning of the data item, etc.

The data space for a module is defined by the set

$$I_s \cup O_s \cup \bigcup_{i=1}^n I_{C_i} \cup \bigcup_{i=1}^n O_{C_i}$$

where I_S , O_S , and O_{C_i} are the inputs and outputs for the processes of the states shown in figure 29. The data interface between the module S and the supramodule in which S is a submodule is given by the set $I_S \cup O_S$.

This initial definition of the variables will serve as a starting point when the control aspect of each module is addressed in the next step. At this point the only control that has been mentioned is the passing of control from a module to its submodules. The intramodule control is described in the next step.

5.2.4 TRANSITION DIAGRAM (STEP 4)

The transfer of control between states in a module can be represented either with flow charts or by transition diagrams. The transition diagram is used to show the flow of control in this methodology because it is felt that it provides better visibility into how the process relates to events in time.

Since the intent is to generate a design that is amenable to structured programming, the transition diagrams will be drawn using a limited set of single-entry/single-exit constructs. Figure 30 shows the set of constructs.

The states that are the submodules in the module are the processes that must be related by some control structure. These control paths are shown by the transitions in the diagram. The transitions are drawn by considering the states in the module two at a time to determine if a transition should occur between the states.

When it is determined that a transition should occur, a condition is defined if needed. This condition is recorded in a table of conditions for this module. The inputs that caused this condition are then determined. These inputs will consist of data items that are recorded in the data-space table for the module. This table's definition was started in step 3. As new entries are determined, they are placed in the table.

A state in a transition diagram represents a process that will be implemented in software. Figure 31 shows a state S and defines (1) inputs and outputs of the process represented by S , (2) conditions that cause transition from a state, and (3) data-space elements used to compute the condition.

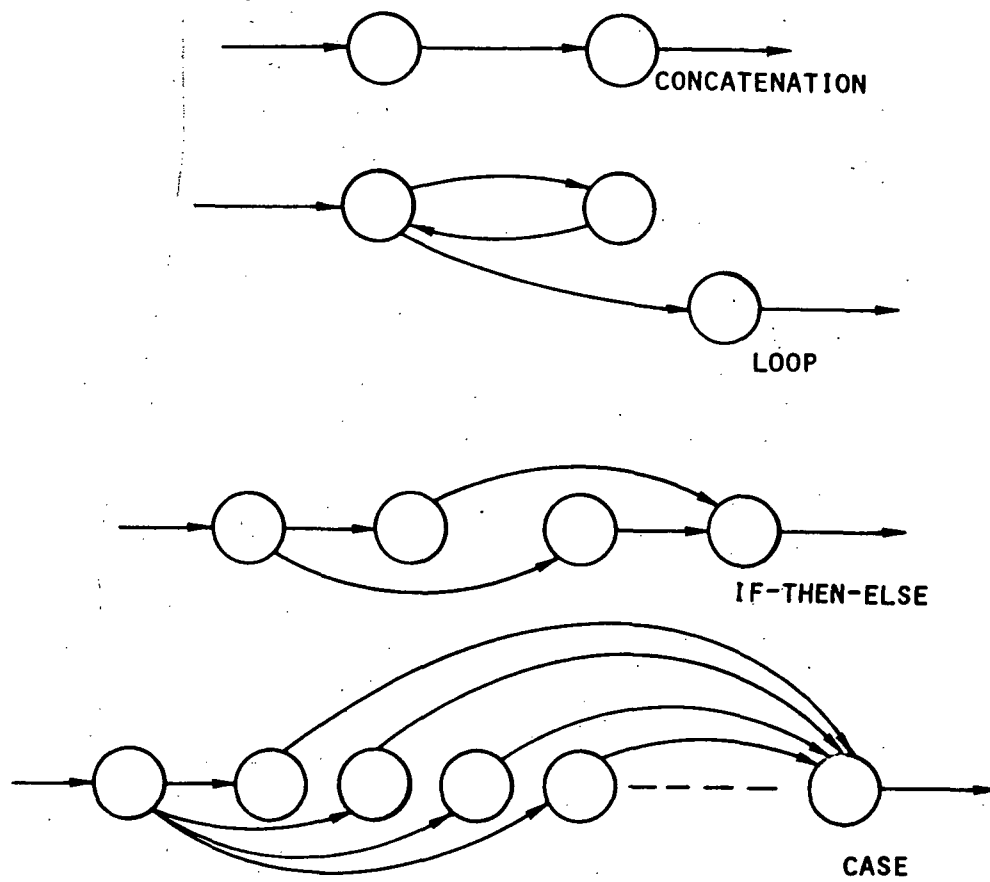
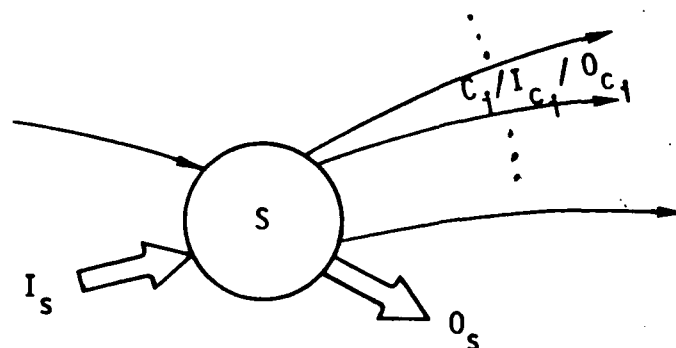
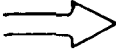



Figure 30.—Software Constructs



I_s	Input to the process P_s
O_s	Output from the process P_s
	Inputs to or outputs from a state
C_i	Condition that causes transition i to be traversed. If a minus sign precedes C_i , then the condition is the complement of C_i .
I_{C_i}	The inputs that caused C_i to be true
O_{C_i}	The outputs that were computed from I_{C_i}
	Represents the transitions into or out of a state
S	The state S that defines a process P_s such that

$$O_s = P_s(I_s)$$

Note that the I_{C_i} are subsets of I_s and the O_{C_i} are subsets of O_s .

Figure 31.—State Definitions

5.2.5 SOFTWARE CODE (STEP 5)

To simplify the coding, a higher level language could be used that supports the constructs and represents a suitable method for handling data structures at the bit, byte, and word level.

Each module can be considered as a single-entry/single-exit block of code. The control structure of the module is coded by using the defined constructs. It is the programmer's responsibility to determine how each module should be implemented. There are two alternatives. One is to use a subroutine; the other is to use "in-line code."

Documentation must proceed with the design and in all cases ultimately describe the software that implements the design. The design is described by documenting each module of the software structure tree. The outline for the module documentation is as follows (see app. B for illustration).

Module: STRUCTURED NAME: Descriptive Name

- | | |
|--|---|
| a) Description: | Paragraph that describes the module's function and references to the requirements, if applicable. |
| b) Subprocesses: | A list of the submodules in the module by short name and descriptive name. |
| c) Inputs and outputs of the subprocesses: | A table of inputs and outputs for each submodule. |
| d) Data-space elements | Described by giving a short name and a description of the element. |
| e) Control flow diagram: | Described by using transition diagrams or flow charts. |
| f) Condition table: | Table that defines the conditions used in specifying the control flow. |

5.3 HARDWARE DESIGN

The ARCS concept emphasizes software processes in achieving fault-tolerant capabilities. The ARCS candidate hardware must be viewed as a vehicle to facilitate an effective software design for the overall reconfiguration and application processes described or identified in the previous sections.

The hardware description is divided into the hardware system architecture, the hardware interfaces, and the computer unit discussed below. The instruction list is presented in appendix C. A detailed discussion of the hardware configuration rationales and trade studies is contained in appendix D.

5.3.1 HARDWARE SYSTEM ARCHITECTURE

The term “architecture” as used here refers to these aspects of the candidate ARCS hardware: the interconnection of the major hardware elements within the system, the functional organization of the computer unit, and the overall redundancy management/reconfiguration structure. Figures 32, 33, and 34 contain block diagrams corresponding to each of these architectural aspects. These diagrams provide the foundation for the discussion of system architecture that follows.

The interconnection of the candidate ARCS hardware is shown in figure 32. The baseline structure is a triplex system with a single computer unit per channel. Each computer unit contains a processor, memory, and all channel interface electronics. Sensor, mode control, and servo interfaces are dedicated on a channel basis, with data exchanged between computers via cross-channel data buses. Each computer exclusively controls the engagement and shut-down on its own servos.

All cross-channel communication (excluding frame sync discretes) is accomplished via dedicated one-way optical serial digital data buses that independently interconnect each computer to each other computer. A functionally separate utility interface with the system test panel is provided on a per-channel basis. This is a serial digital interface per EIA Standard RS-232C. The interfaces required for an expansion to a quadruplex redundancy level are shown by the broken lines in figure 32.

A functional block diagram of the ARCS computer unit is illustrated in figure 33. Key features include the following:

- Architecture is highly bus organized.
- Flexible computer I/O system uses the “directly operable input/output” (DOIO) concept, which:
 - Provides direct memory access (DMA) input/output capability as needed by each interface element, without interference with CPU operations
 - Provides interrupt-initiated or processor-controlled input/output using a standard device-controller structure
 - Allows the CPU to access and operate on all I/O data directly within the RAM memory structure
- Solid-state memory is partitioned into functionally independent program memory (PROM) and variable/scratch-pad memory (RAM) sections.

The redundancy management block diagram for each channel of the ARCS is shown in figure 34. Key features of the reconfiguration design indicated by this diagram (in conjunction with the previous two diagrams) are the following:

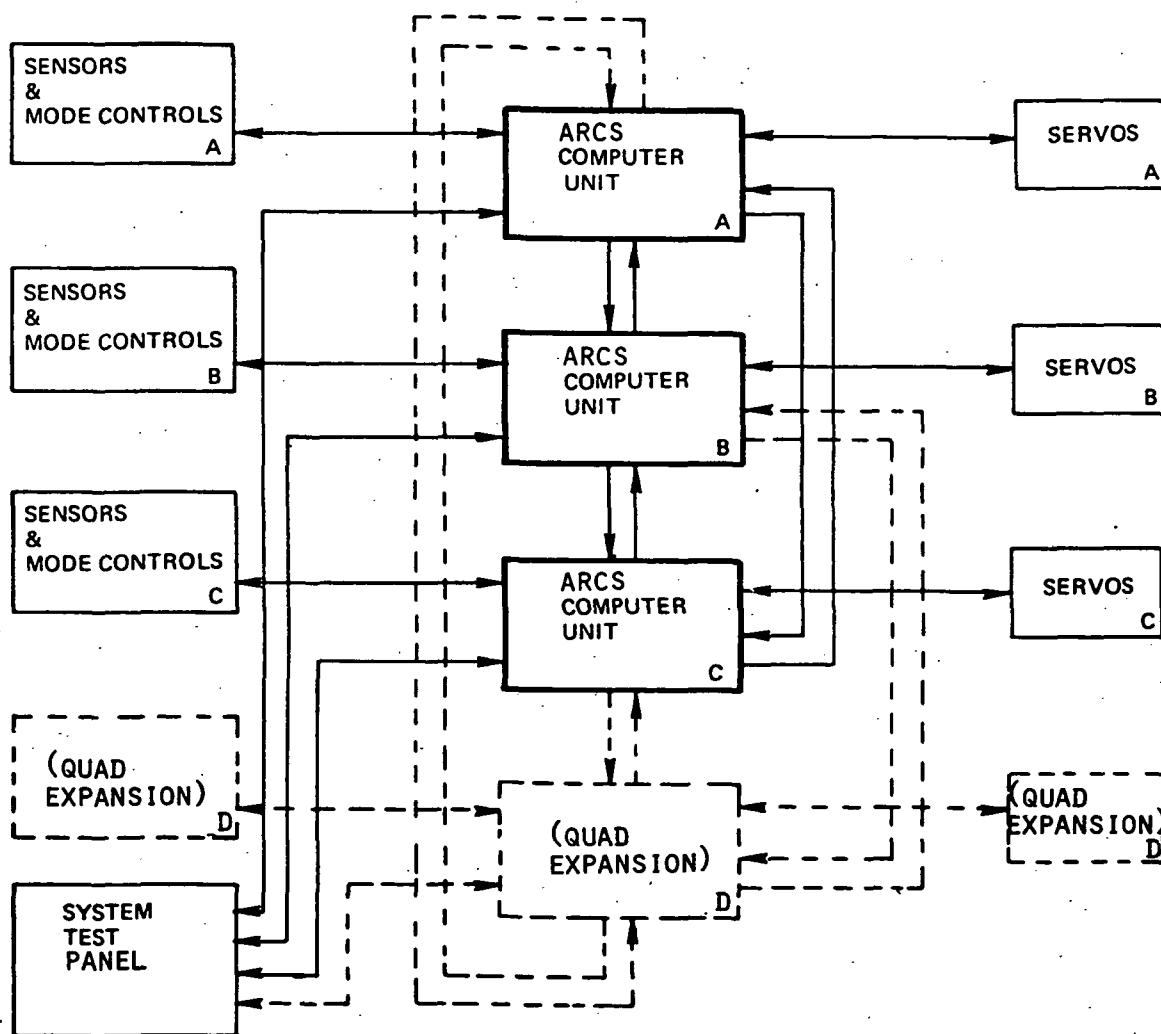


Figure 32.—ARCS Hardware Interconnection Block Diagram

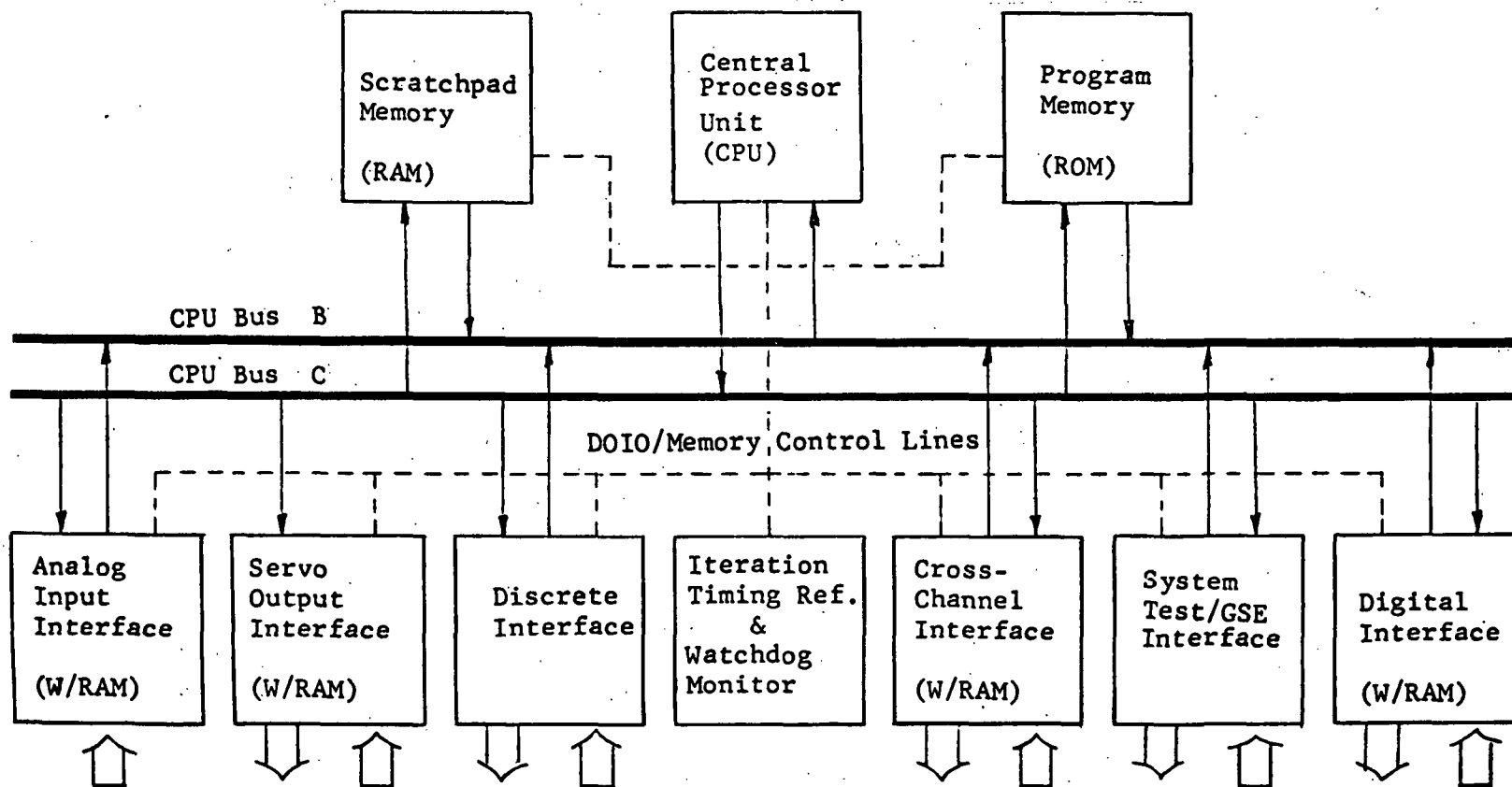


Figure 33.—ARCS Computer Unit Block Diagram

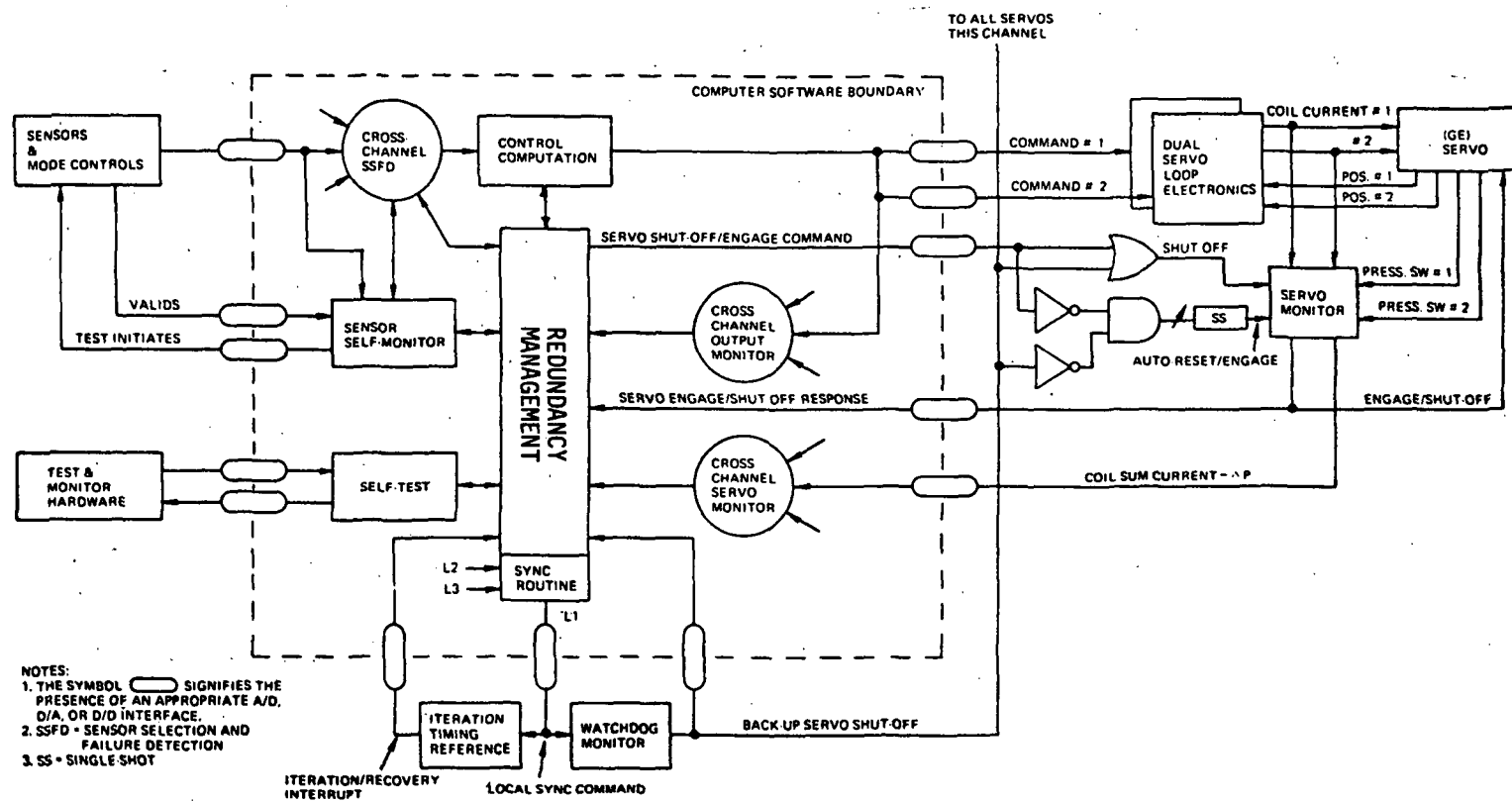


Figure 34.—ARCS Redundancy Management Block Diagram

- Triplex architecture is based on two voting nodes, one at the level of sensor data and the other at the force-summing point connecting servoactuator outputs.
- With appropriate operating software, the system tolerates all first failures, a high percentage of second like failures, and multiple numbers of transient faults at each voting node.
- Frame-synchronous computation is achieved through a software sync routine using cross-channel discretes.
- Iteration timing reference provides real-time interrupt and, in conjunction with the operation of the sync routine and the watchdog monitor, also initiates recovery operations.
- Software servo shutdown decisions are backed up with two hardware monitors in each computer unit: the watchdog monitor and the dual servo-loop electronics monitor (one per servo).
- Dual servo-loop electronics are used to provide servo self-monitoring capability for all faults not detectable by cross-channel comparison.
- The watchdog monitor detects gross computer failures that cause loss of real-time control.
- All cross-channel comparisons of computer outputs and servo differential errors are accomplished in software.
- All cross-channel data exchange is initiated by the sender.

5.3.2 HARDWARE SYSTEM INTERFACES

The ARCS hardware utilizes a software sync routine to establish and maintain frame-synchronous computations in all three computer channels. To facilitate this type of operation, each channel issues a local sync command (LSC) discrete that has these functions:

- When SET, it informs the other channels that the local channel is ready to sync.
- When CLEARED, it causes the iteration timing reference counter to reset and begin counting out the next iteration period.
- When not SET and CLEARED in a real-time period that falls within a time interval tolerance, it causes the watchdog monitor to indicate computer failure.

As indicated by figure 35, the LSC discretes are cross-channel interchanged as bits in each computer's status register. The iteration timing reference generates a priority interrupt that defines the real-time frame rate. If the LSC fails to satisfy the requirements of the watchdog monitor, then a computer failure is indicated, and the next interrupt from the iteration timing reference is interpreted as a recovery interrupt by the software. The integration of the

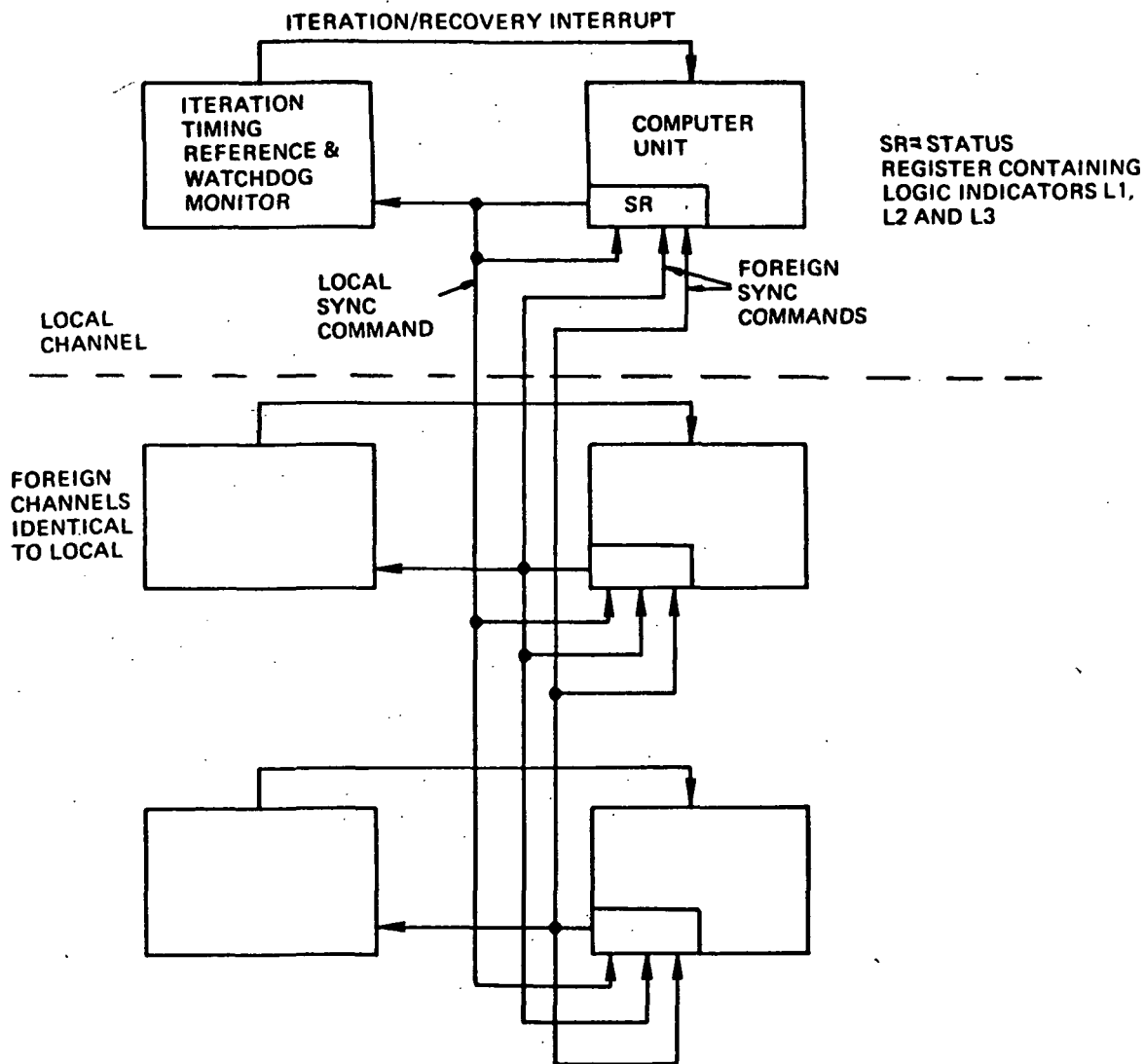


Figure 35.—Interface to Allow Software Frame Synchronization

iteration and recovery interrupts into a single interrupt minimizes the additional hardware that must be provided to facilitate recovery within ARCS. In addition, it provides a regular exercise, and therefore validation, of nearly all of the electronics involved in initiating recovery from outside the processor.

The ARCS uses cross-channel data exchange to achieve many of the fault-tolerant characteristics. A high-speed, one-way, serial, digital, optical data link allows the following data transfers from each computer to any other computer:

- Sensor data for sensor selection and failure monitoring
- Output commands for failure monitoring
- Servo data for failure monitoring
- Recovery data
- Failure status and maintenance data

The use of an optical data link ensures complete electrical isolation between computers.

Figure 36 illustrates the basic data link structure between any two channels. Each transmitter operates in the processor-controlled DOIO mode. Data is sent in bursts of up to 64 words each. A total of 1024 words of dedicated RAM storage is provided at each receiver. The receivers operate in the DMA mode of the DOIO system. Transmissions are at a 2-MHz bit rate using a self-clocking signal format from MIL-STD-1553.

The triplex force-summed servoactuator design for ARCS combines experience from two fly-by-wire systems by using:

- Low-pressure-gain servovalve/servoactuator modules and cross-channel monitoring concepts from the 680J secondary actuator (Survivable Flight Control System Development).
- Self-monitored, independent channel concepts from the HLH-DELS Program.

The selected design has the distinct advantage of not requiring differential pressure feedback equalization. Figure 37 shows a block diagram of the servoelectronics/servoactuator concept.

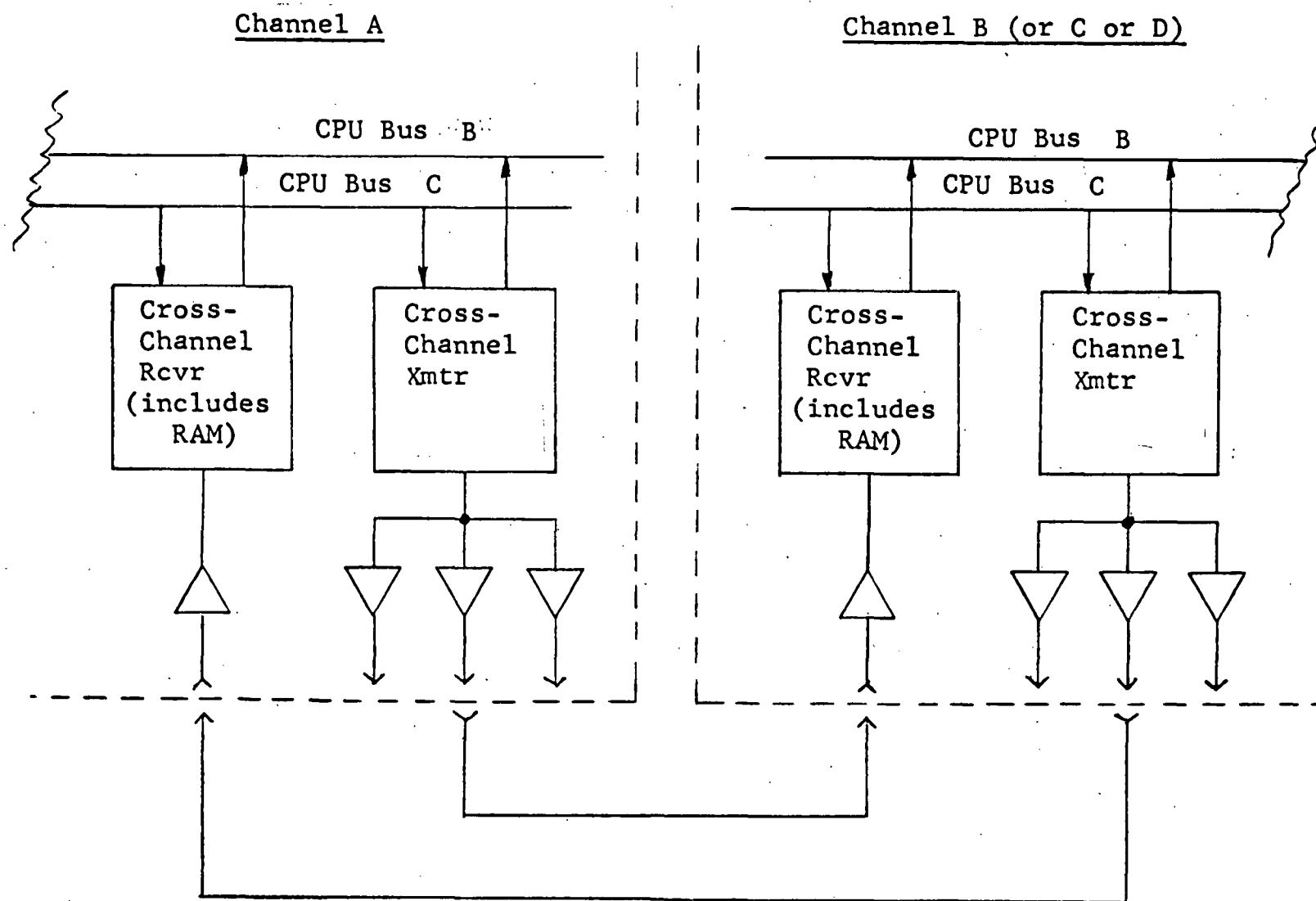


Figure 36.—Cross-Channel Data Link Structure

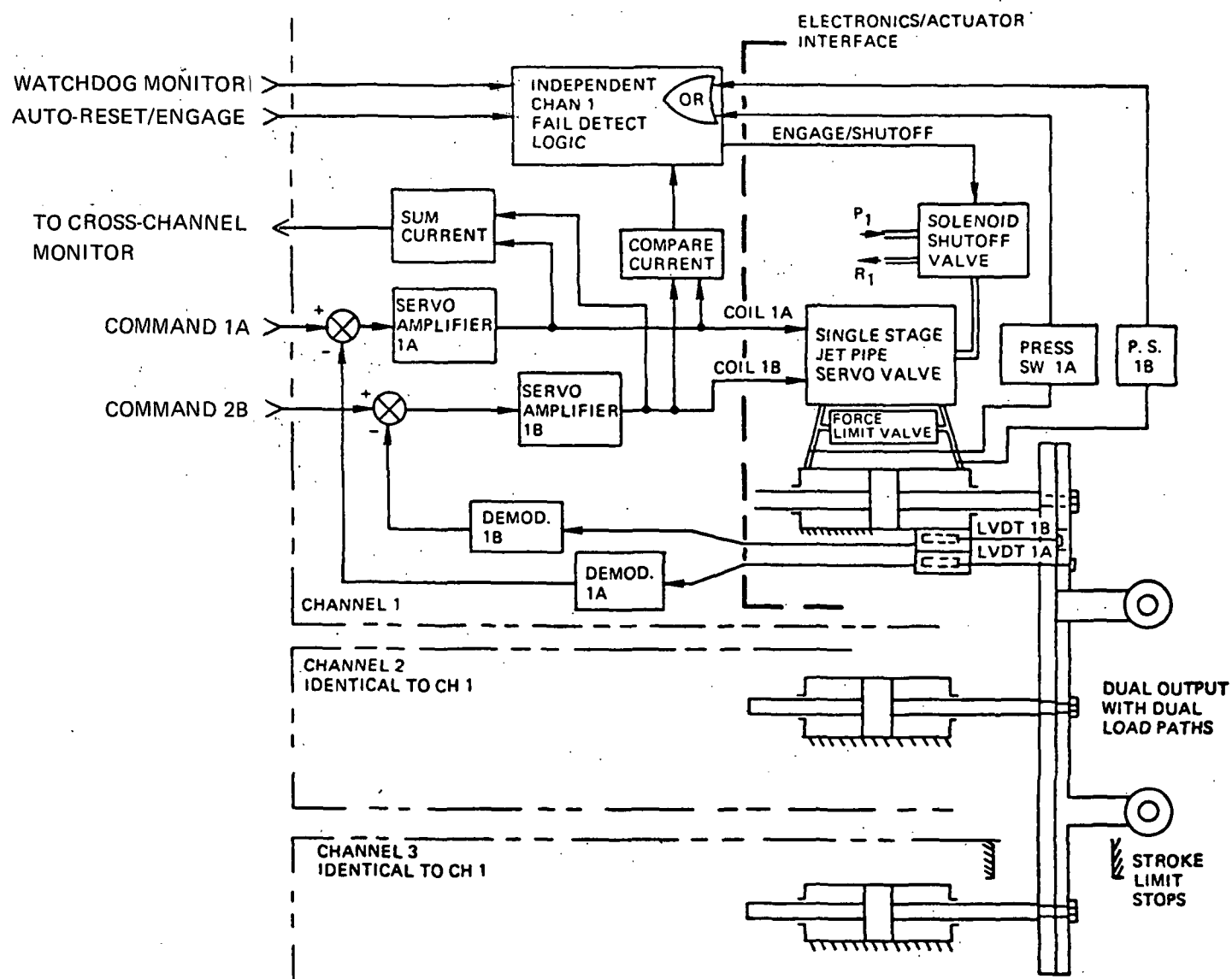


Figure 37.—Servoelectronics/Actuator Concept

Since each servo uses dual servo-loop electronics for the purpose of self-monitoring, each computer must generate two D/A output commands per servo. In addition, a servo engage/shutoff discrete is required for each servo. The computer software control of this discrete includes the capability to automatically reset and reengage a bypassed servo. Figure 37 outlines this operation. Notice that automatic reengagement is interlocked with the output of the watchdog monitor. This monitor must indicate that the computer is not failed before an automatic reset and reengage can occur. On the other hand, either the computer-generated servo engage/shutoff discrete or the watchdog monitor backup shutoff discrete can cause servo bypass.

The servos are monitored by two different techniques:

- Comparison of servovalve dual-coil currents (differential current) within the dual servo-loop electronics
- Cross-channel comparison of servovalve (sum) coil current (which is proportional to differential pressure) using computer software

The first technique is used to detect failures within the servo-loop electronics and the position feedback sensors. The second technique is used to detect both servo and computer command failures that are not detected by other monitoring functions. In addition, pressure switches are used to detect gross pressure differentials (hardover errors) or loss of hydraulics. This combination of servo monitoring techniques makes it possible to achieve two-fail-operational performance for a large proportion of second failures.

The D/A outputs from the processor are generated in the processor-controlled DOIO mode using what appears as a dedicated D/A per output. In effect, this means that no analog sample-holds limitations apply. Individual outputs may be “frozen” or updated at different rates as determined by the software. Each D/A output appears as one RAM memory location to the processor.

Within the ARCS hardware architecture, input sensors and mode controls are dedicated on a channel basis. Thus, there is no cross-strapping of sensor data prior to the analog-to-digital and digital-to-digital interfaces. All cross-channel exchange of sensor data is accomplished via the cross-channel data link.

The control of sensor input conversions occurs within the DOIO structure. For analog inputs, a multiplexed A/D converter begins its sample sequence following the iteration interrupt. Inputs are converted and stored in RAM memory in a DMA mode.

Because of the DOIO structure, RAM memory for DMA operations is accessed with no interruption of CPU activity. Once stored in RAM, the input data is immediately available for use by the program. Double buffering of analog data for (bit-identical) sensor selection purposes is not required with appropriate executive scheduling of computations.

Serial digital inputs, from asynchronous external devices, are received using dedicated standard interface receivers with one receiver per data source. Input data is stored in RAM

using the DOIO DMA mode. A fixed block of RAM storage is assigned to each serial digital receiver. Double buffering for (bit-identical) sensor selection is required.

Serial digital outputs to asynchronous external devices (excluding the RS-232C system test interface) are provided using dedicated standard interface transmitters. One transmitter is used for each different output standard. Each transmitter operates in a processor-controlled mode and, within DOIO, appears as one RAM memory location to the processor.

Discrete I/O is handled in a DMA mode for inputs and a processor-controlled mode for outputs. Discretes are packed within the 16-bit data word format and, with the bit manipulation instructions, are easily operated on by the program.

5.3.3 ARCS COMPUTER UNIT

The functional block diagram of the ARCS computer unit was presented in figure 33. The major functional elements shown in the block diagram are the following:

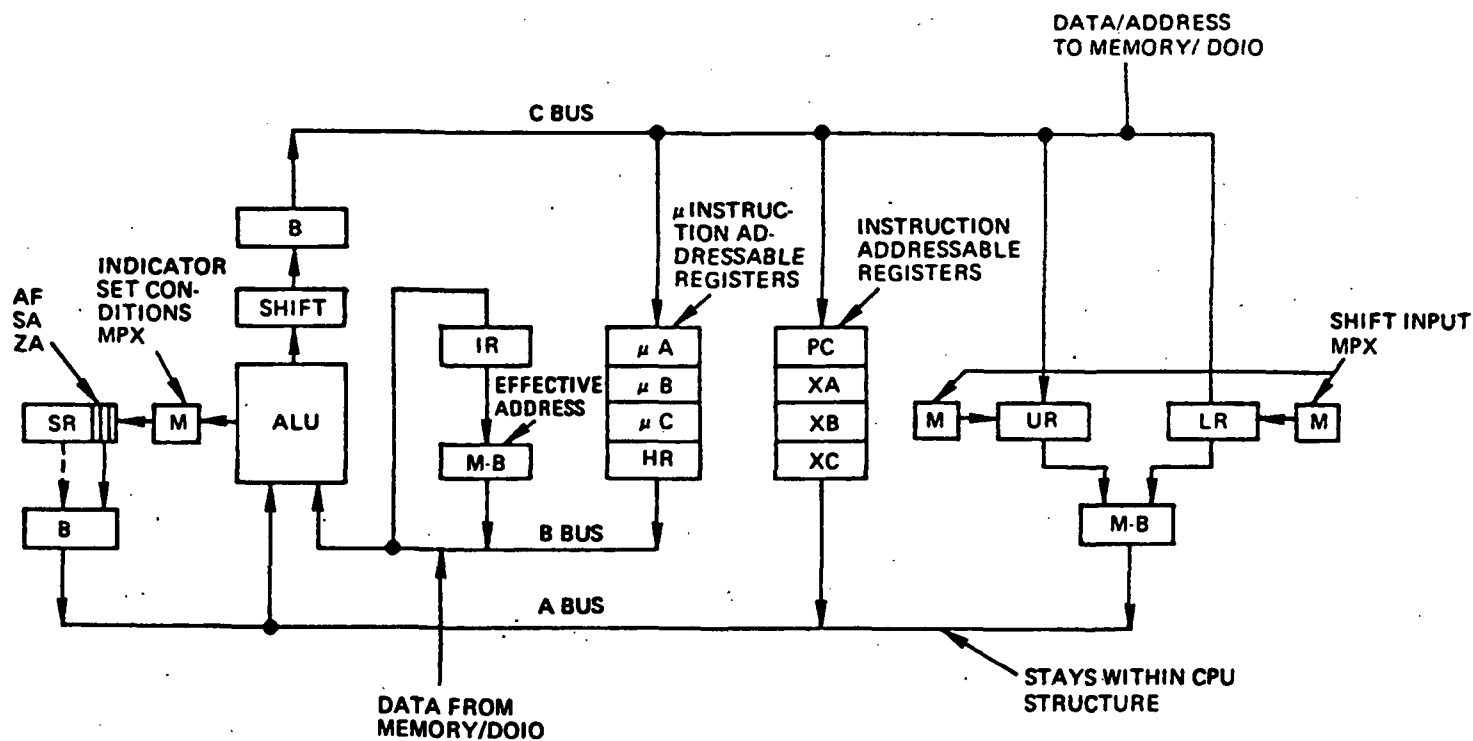
- Central processor unit
- Program memory
- Scratch-pad (variable) memory
- Analog input interface
- Servo output interface
- Discrete interface
- Iteration timing reference and watchdog monitor
- Cross-channel interface
- System test/GSE interface
- Digital interface

Each of these functional elements is described below, along with power supply and test and monitoring functions.

5.3.3.1 CPU and Memory

Table 2 summarizes the functional characteristics of the processor.

The CPU interfaces with and controls the program memory, the scratch-pad memory, and all the interface elements within the directly operable input/output (DOIO) system. The CPU receives instructions from program memory, interprets them, and performs the indicated arithmetic, logical, branching, or control operation. The process of memory access and



AF = ARITHMETIC OVERFLOW
 SA = SIGN OF ADDER
 ZA = ADDER ZERO
 UR = UPPER REGISTER
 LR = LOWER REGISTER
 PC = PROGRAM COUNTER

IR = INSTRUCTION REGISTER
 HR = HOLDING REGISTER
 μA = μ -PROGRAM REGISTER A
 μB = μ -PROGRAM REGISTER B
 SR = STATUS REGISTER
 XA = INDEX REGISTER A

XB = INDEX REGISTER B
 XC = INDEX REGISTER C
 ALU = ARITHMETIC LOGIC UNIT
 μC = μ -PROGRAM REGISTER C
 B = BUFFER
 MPX, M = MULTIPLEX

Figure 38.—Processor Register Structure

Table 2.—Functional Characteristics Summary of ARCS Processor

Item	Characteristics
Type	General purpose, stored program, uniprocessor
Number system	Binary, fixed-point, 2's complement, fractional
Data word length	16 bits standard, 32 bits double-precision
Instruction word length	16 bits
Register structure	Accumulator organized with three index registers
Instructions	Microprogrammed set of 104 with application-dependent spares for special op-codes
Throughput	420 kops (85% add, 10% multiply, and 5% divide)
Address modes	Direct, indirect, program counter and index register relative, and immediate
Interrupts	Eight level, software maskable
Memory structure	Independent program and variable memories
Input/output structure	DOIO concept: allows interrupt-controlled, processor-controlled, or noninterference DMA-controlled input/output

instruction execution is controlled by a stored microprogram within the CPU. Because of this microprogrammable design, the CPU may be conveniently divided into two distinct functional elements: the microprogram control structure and the register structure. It is the microprogram control structure that maps each machine op-code into particular activity within the register structure.

The register structure shown in figure 38 is a three-bus system centered about the arithmetic element. All buses are 16 bits in width. The A and B buses are tristate so that multiple sources may be enabled onto them. The B and C buses provide the communication links for address and data for program memory, scratch-pad memory, and input/output devices. The A bus is resident within the CPU register structure.

The instruction set is tailored for real-time control applications. It has a full complement of load/store, arithmetic double precision, logical, branching, register, input/output, and shifting instruction that have proven useful in control applications. The instruction set is an expansion of the General Electric MCP-701 instruction set, particularly in the area of immediate instructions, bit manipulation instructions, interrupt handling instructions, and the scratch-pad memory reference instructions that allow direct manipulation of input/output data. In addition, the processor has reserved 28 instructions that may be microprogrammed to suit the application. All instruction execution times are based on a 1.0-us cycle time program memory. The detailed instruction-by-instruction description is presented in appendix C.

The CPU has an eight-level priority interrupt system. The highest level is not maskable whereas the seven lower levels are maskable using the seven most significant bits of the status register. All interrupts are set by external stimuli and reset by the software. The priority of the seven maskable interrupts is established by a linear daisy chain. This priority is alterable under software control by using the mask bits of the status register. The processor has four machine language instructions specifically designed to minimize the overhead burden associated with interrupt processing.

The program memory is separate and independent from the variable scratch-pad memory. Interfacing between a memory module and the CPU is by way of the B and C buses. The CPU supplies control lines to all memory modules. A memory module is active only if the address supplied from the CPU enables it. The program memory is composed of two sections:

- PROM (fusible link) memory
- Nonvolatile storage zone

Because of the flight-critical nature of fly-by-wire applications, the memory configuration provides for maximum software control and the best resistance to transient conditions that may affect destructive readout (DRO) configurations. It employs semiconductor read-only or programmable read-only memories for program store.

A nonvolatile storage zone is provided to record and hold a historical record of fault occurrences for engineering and/or shop maintenance purposes. This portion of program memory is therefore associated exclusively with the maintenance aspects of the ARCS system test function.

The processor has semiconductor scratch-pad random access memory for intermediate variable storage and bulk variable storage. It has its own memory address register and enable and control logic. It is controlled by the CPU independently from the program memory.

5.3.3.2 Directly Operable Input/Output (DOIO)

Input/output devices are imbedded within the scratch-pad memory addressing structure. The data from the input/output device may be directly operated on rather than first transferring it to main-line variable memory. DOIO treats input/output data as distributed memory locations. Each distributed memory section is a functionally independent memory. Each additional input/output device contains its own scratch-pad area.

Three types of input/output sequences are permitted:

- DVC—device-controller interrupt initiated
- DMA—direct memory access to the distributed scratch-pad
- PC—processor controlled

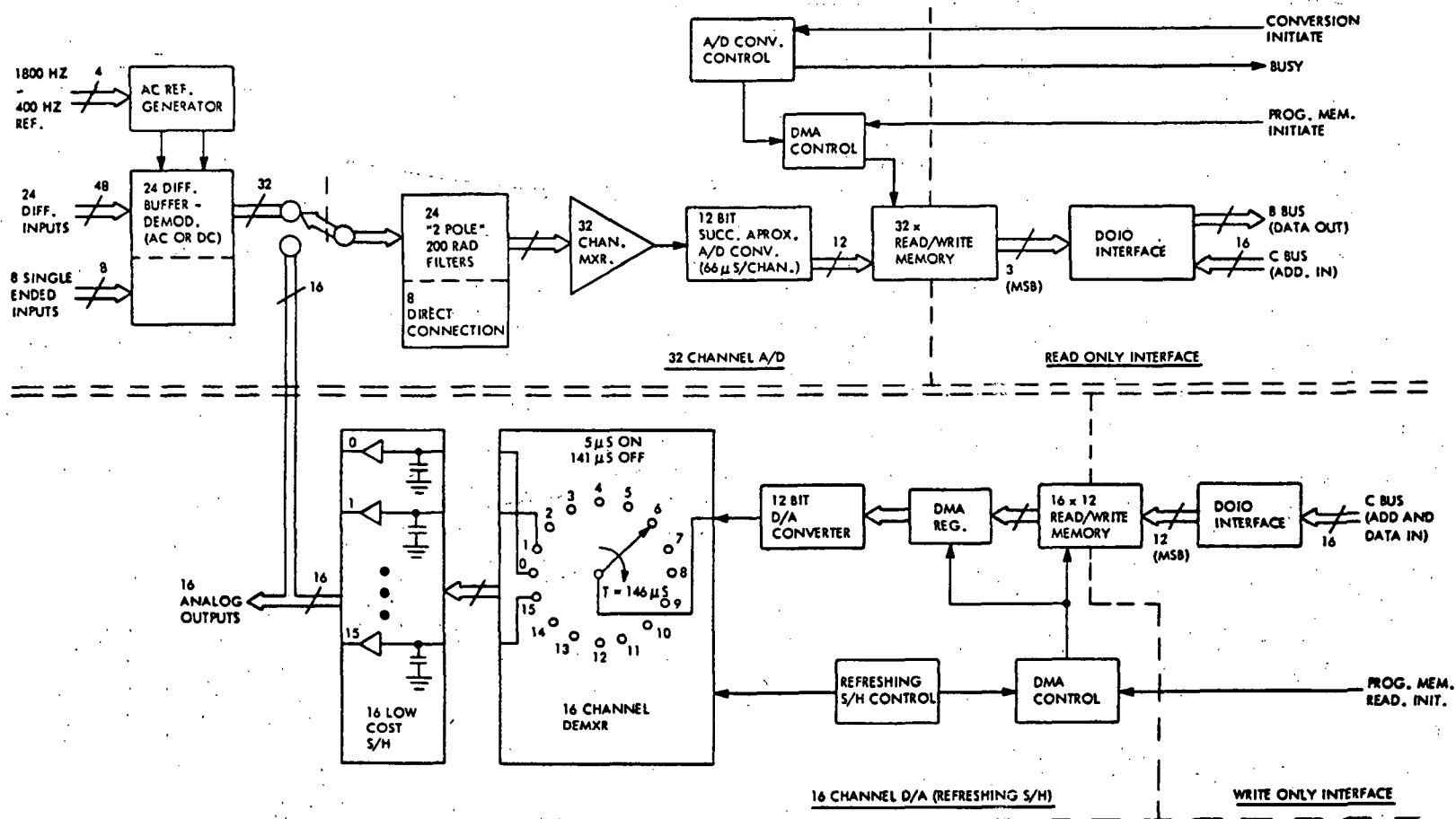


Figure 39.—Combined Analog I/O Module

The complement of input/output functions used within the ARCS computer unit is:

- Analog input conversion and conditioning circuits
- Discrete I/O
- Analog outputs and servoamplifiers
- Serial digital sensor interface
- System test interface
- Cross-channel data interface

In addition to these input/output functions, the iteration timing reference, watchdog monitor, and servo disengage logic are embedded in the functional input/output area.

The processing of analog inputs is combined on a standard module with an analog output section. Figure 39 shows a block diagram of this combined analog I/O module.

The analog input interface contains a 32-channel multiplexed A/D converter, 24 ac/dc signal conditioners, and 8 direct dc inputs. Input conversions are initiated by a CPU-controlled discrete. Each time the conversion initiate line is pulsed, the input multiplexer sequences 32 analog inputs through the A/D.

The digital results are then stored in the local scratch-pad memory in a DMA mode. Within the DOIO structure, direct memory access to scratch-pad memory is possible every time the CPU accesses the program memory. A minimum access rate is guaranteed for the normal instruction set by the microprogram control of the program memory initiate line. The CPU may access the local scratch-pad memory for A/D data at any time without experiencing DMA interference.

The servo output interface is composed of two parts: the output D/A section and the servoelectronics section. A block diagram of the output D/A section is contained in the combined analog I/O module diagram in figure 39.

A block diagram of the discrete I/O interface is shown in figure 40. Within the DOIO structure, discrete inputs appear as individual bits within four dedicated scratch-pad memory locations. Input level changers are provided to receive up to thirty-two 28-Vdc discretes. In addition, there is provision for twenty-four 5-Vdc transistor-transistor logic (TTL) discrete inputs. Each scratch-pad memory read access causes the input discretes within the selected word to be enabled onto the B bus. The system software is responsible for any contact debounce processing or double buffering within a frame time.

Discrete outputs are generated simply by storing each discrete within a particular bit position in one of four dedicated scratch-pad memory locations. Output level changers are provided for up to eight 28-Vdc discretes. A total of thirty-two 5-Vdc TTL output discretes are provided.

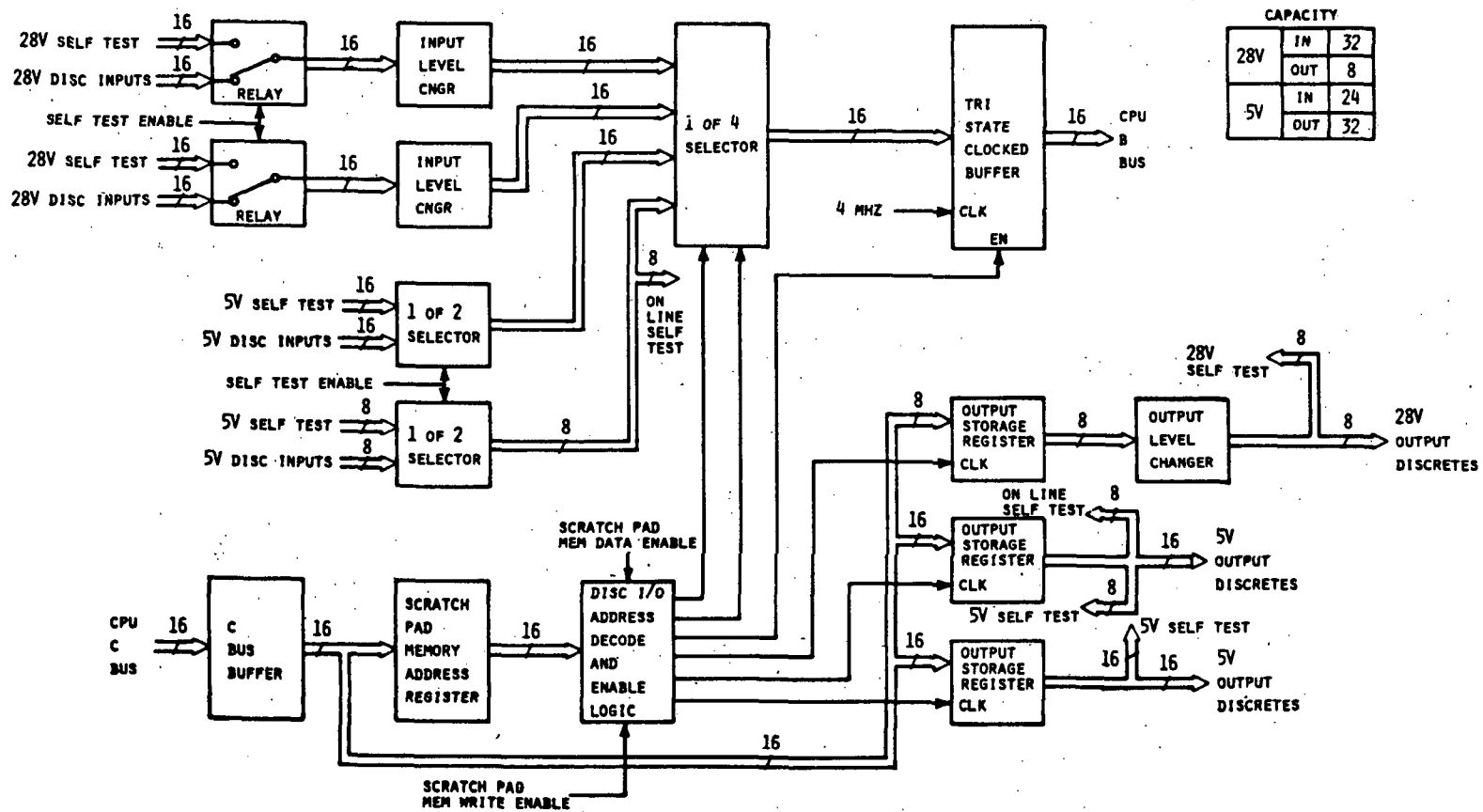


Figure 40.—Discrete I/O Interface

A utility interface defined by EIA Standard RS-232C is provided for use by the system test panel and any ground support equipment. A block diagram of this interface is shown in figure 41. A single UART (universal asynchronous receiver/transmitter) device provides the serial-to-parallel/parallel-to-serial conversions and the signal formatting necessary for the RS-232C serial data link.

A transmission is initiated by the CPU first storing an eight-bit byte (low-order eight bits) from the C bus in the scratch-pad memory address dedicated to the transmitter and then executing a CLR instruction for that address. When serial transmission is complete, an interrupt is generated. The CPU may then transmit another eight-bit byte or mask the interrupt.

Data is received through the UART and an interrupt is generated. The CPU accesses the data by performing a read operation on the scratch-pad memory address dedicated to the receiver. Data is received in eight-bit bytes and enabled onto the low-order eight bits of the B bus when read. The CPU may mask the receiver interrupts at any time.

5.3.3.3 Cross-Channel Interfaces

The cross-channel data link is used to exchange data between computers for normal mode operations such as sensor selection and output monitoring and for state variable data needed for a recovery attempt by one of the channels. The cross-channel receiver contains a 1024-word RAM that is considered large enough so that no double buffering of any cross-channel data is required. Each word in the receiving buffer has a unique definition and it may be processed directly from the buffer. The ability to leave the data in the buffer and not move it to allow another word with a different meaning to share the receiver location saves a significant amount of processor time.

The cross-channel transmitter shown in figure 42 contains a 64-word last-in, first-out (LIFO) stack. The stack is composed of a 10-bit label and a 16-bit data word. The CPU loads the stack with the data to be transmitted and then initiates transmission by executing a CLR XMIT instruction (XMIT is the select address of the transmitter). Any number of words from 1 to 64 may be transmitted at one time. If a CLR is executed with no data in the stack, nothing is transmitted. If at least two words are loaded into the LIFO and then a CLR instruction is executed, the receiver will start transmitting. The CPU may continue to store into the LIFO while transmission is progressing; however, no transmission sequence can be guaranteed using this procedure.

When the stack is empty (when the last word has been read into the parallel/serial transmit register and transmission initiated), an interrupt is issued so that the CPU may load the stack with new data. After the last piece of data for a particular sequence is transmitted, the CPU must mask the transmitter to prevent additional interrupts. If the interrupts are masked, the device ready flip-flop may be tested by the CPU to determine if the transmitter LIFO is empty.

The transmitter is selected using a 16-bit address. The high-order 6 bits of the address are the transmitter select address, and the low-order 10 bits are stored into the label portion of the stack associated with the data word. This label is transmitted with the word and is

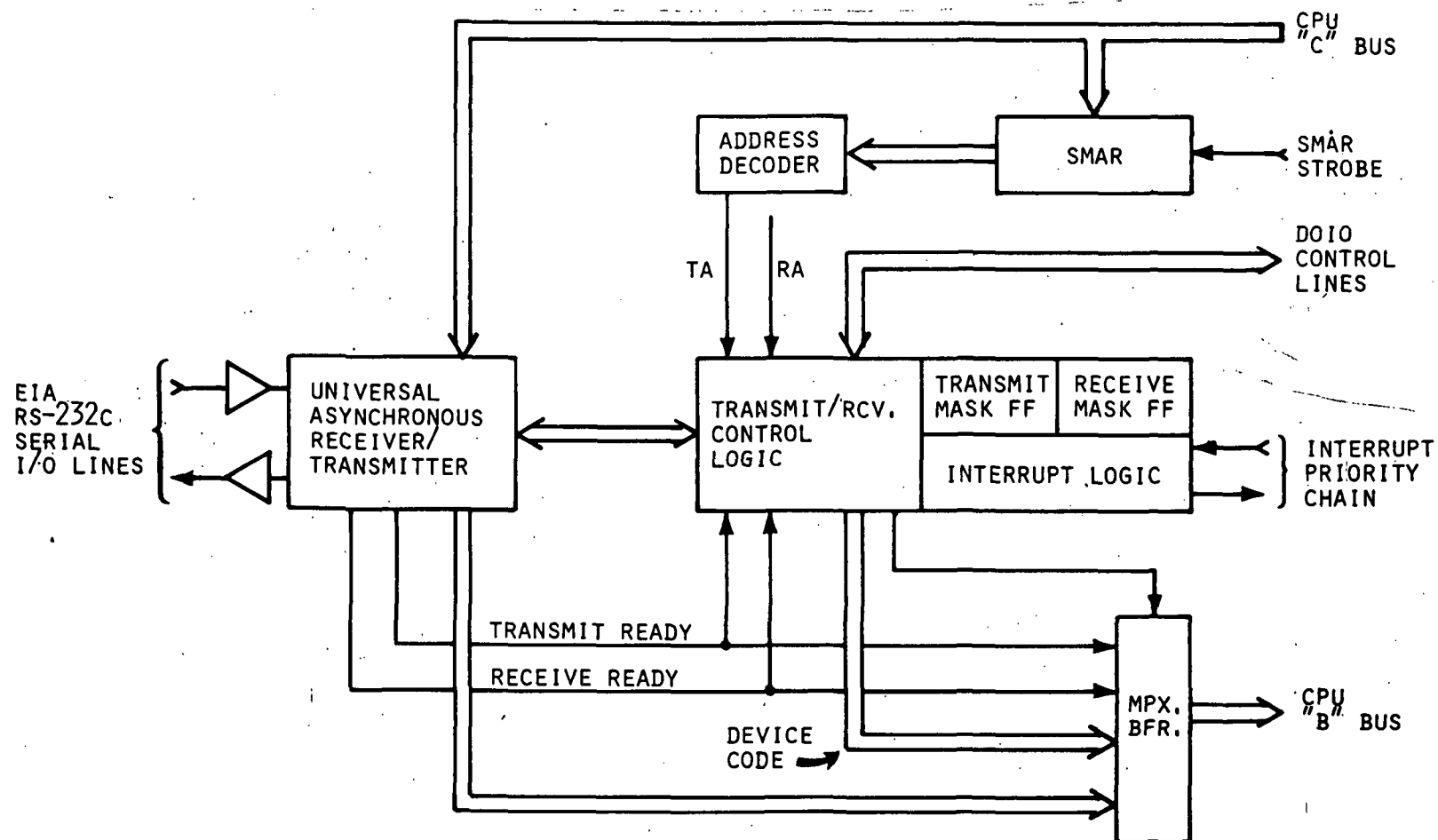


Figure 41.—System Test/GSE Interface

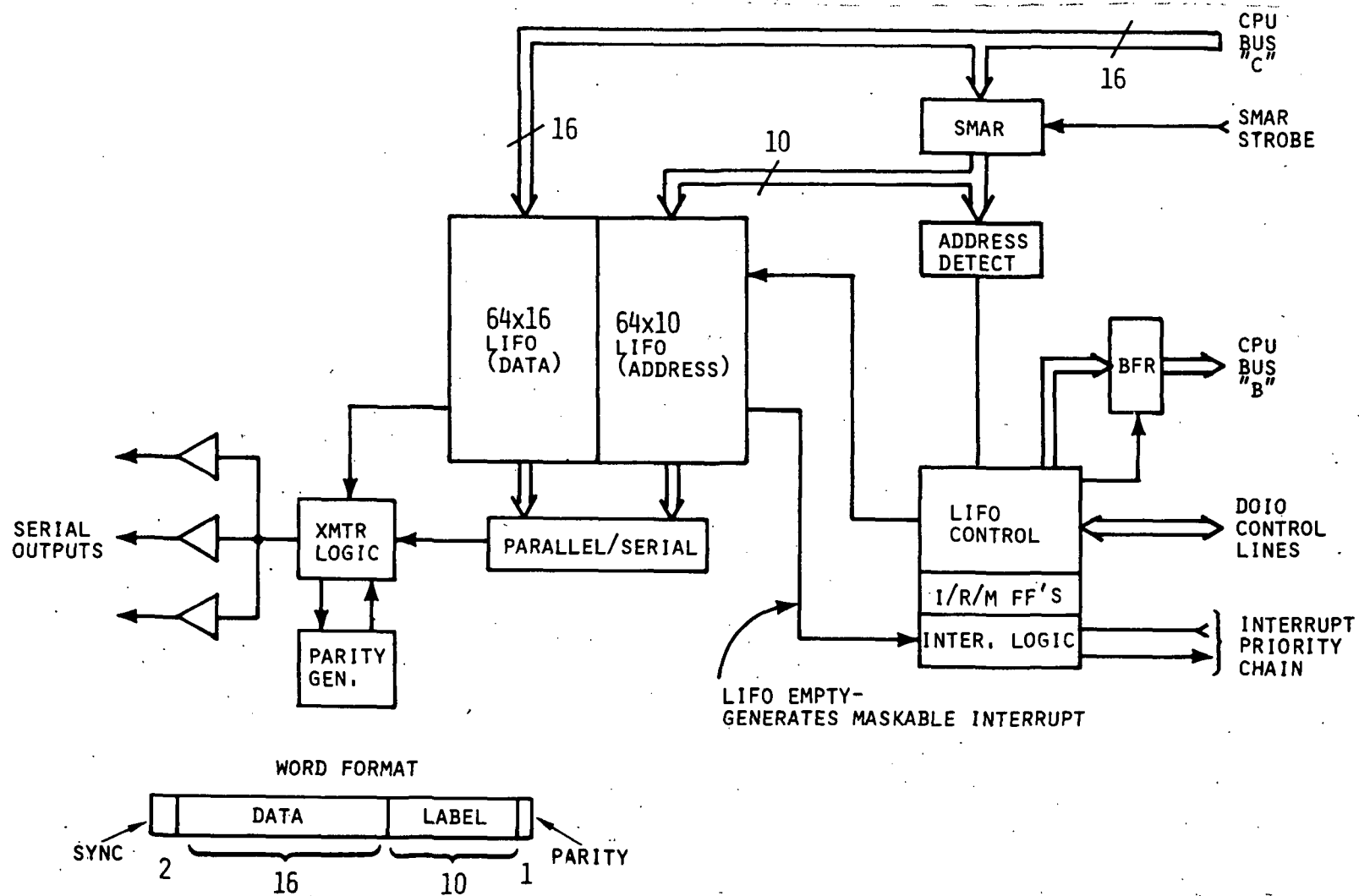


Figure 42.—Cross-Channel Transmitter

used by the receiver as the relative address, within the receiver buffer, to store the data word. More than one word with the same label may be transmitted, with the last one overwriting the previous ones. The transmitter software must be written so that no more than 64 words are stored into the LIFO. Otherwise, the LIFO address pointer will overflow, causing the next word to overwrite the first word stored. When the transmission is initiated, only the number of words indicated by the LIFO pointer will be transmitted. The last word to be stored in the LIFO is pointed to by the LIFO counter. If the CPU attempts to read any location in the stack, this is the word it will read.

Cross-channel data is transmitted at a 2-Mhz bit rate. A total of 29 bits are transmitted: 10 label, 16 data, 1 parity, and 2 for transmitter/receiver synchronization. To ensure that the receiver has time to store the data into three different buffers, there is a 5.0- μ s separation between words. One word is thus transmitted every 19.5 μ s.

A block diagram of the cross-channel receiver is shown in figure 43. Each of the receiver's buffers contain 1K of memory, and each is hard-wired to a particular transmitter. Therefore, no selection process is required to turn on the receiver. When a word is received, the 10-bit label is used as the relative address within the buffer at which the data is stored. The data is stored in a DMA mode. No transmission word can have a label outside the buffer, since the 10-bit label may range from only 0-1023. The receiver buffer memory has both read and write capability, so a word can be modified during processing and stored back onto the same location. The receiver checks the label associated with each transmission, and if it is label zero, an interrupt is generated. The transmitting computer may load label zero at the end of a block of data that corresponds to a particular function. The receiver will generate an interrupt when all of that data has been received. The CPU may then process the data without waiting for the whole block transfer to be completed. Since the transmitter has the capability of loading the LIFO stack with more than one word with label zero, the data in the stack can be separated into blocks. Each time a complete block is received, the CPU gets an interrupt, and it can check the word associated with label zero for a code describing the type of transmission. This interrupt may be masked if not needed or desired. The ready flip-flop may be tested to see if a label zero has been received in the case where the interrupt has been inhibited. If a parity error is detected on the transmission, the receiver sets the word to zero and stores it at label zero. This results in an interrupt being generated. The CPU may choose to ignore the data or the interrupt.

5.3.3.4 Iteration Timing Reference and Watchdog Monitor

The iteration/recovery interrupt is a local channel priority interrupt. It is generated from the iteration timing reference, which is a counter that serves as the real-time reference for the local channel. It may be reset by the local sync command negative transition. This will rezero or reinitialize the count period. If not reset by the local sync command, it will continue to toggle count states and thus generate a recovery interrupt.

The watchdog monitor is composed of two sections. The first section measures the time interval between negative edges of the input signal and generates an output pulse (F) immediately following a negative edge if the previous period is within specified limits, as shown in figure 44. The second section is an ac-coupled monostable multivibrator that will generate a computer-failed discrete (H) after a specified amount of time (1.5 periods) unless reset by an in-tolerance pulse (F).

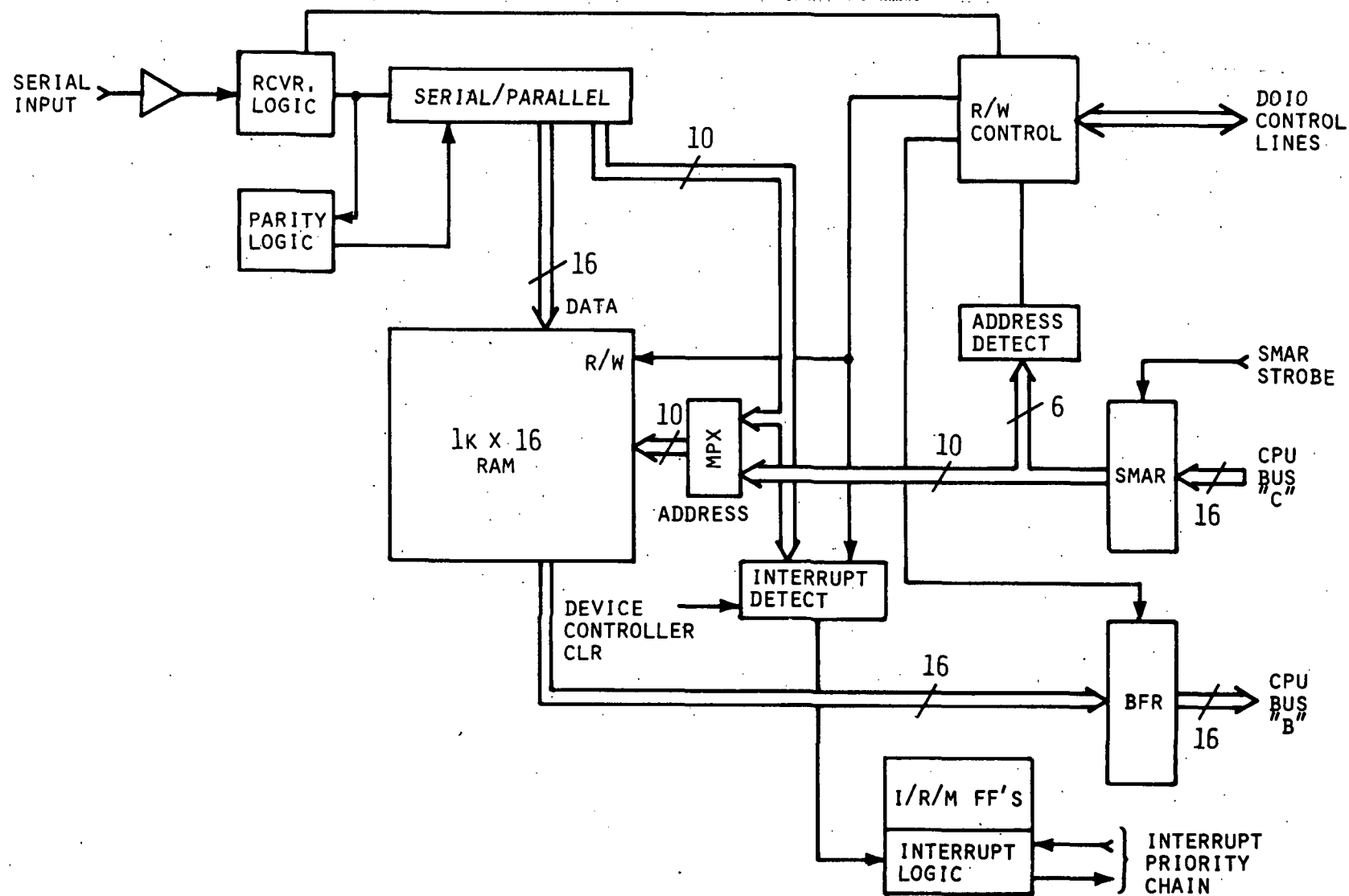


Figure 43.—Cross-Channel Receiver

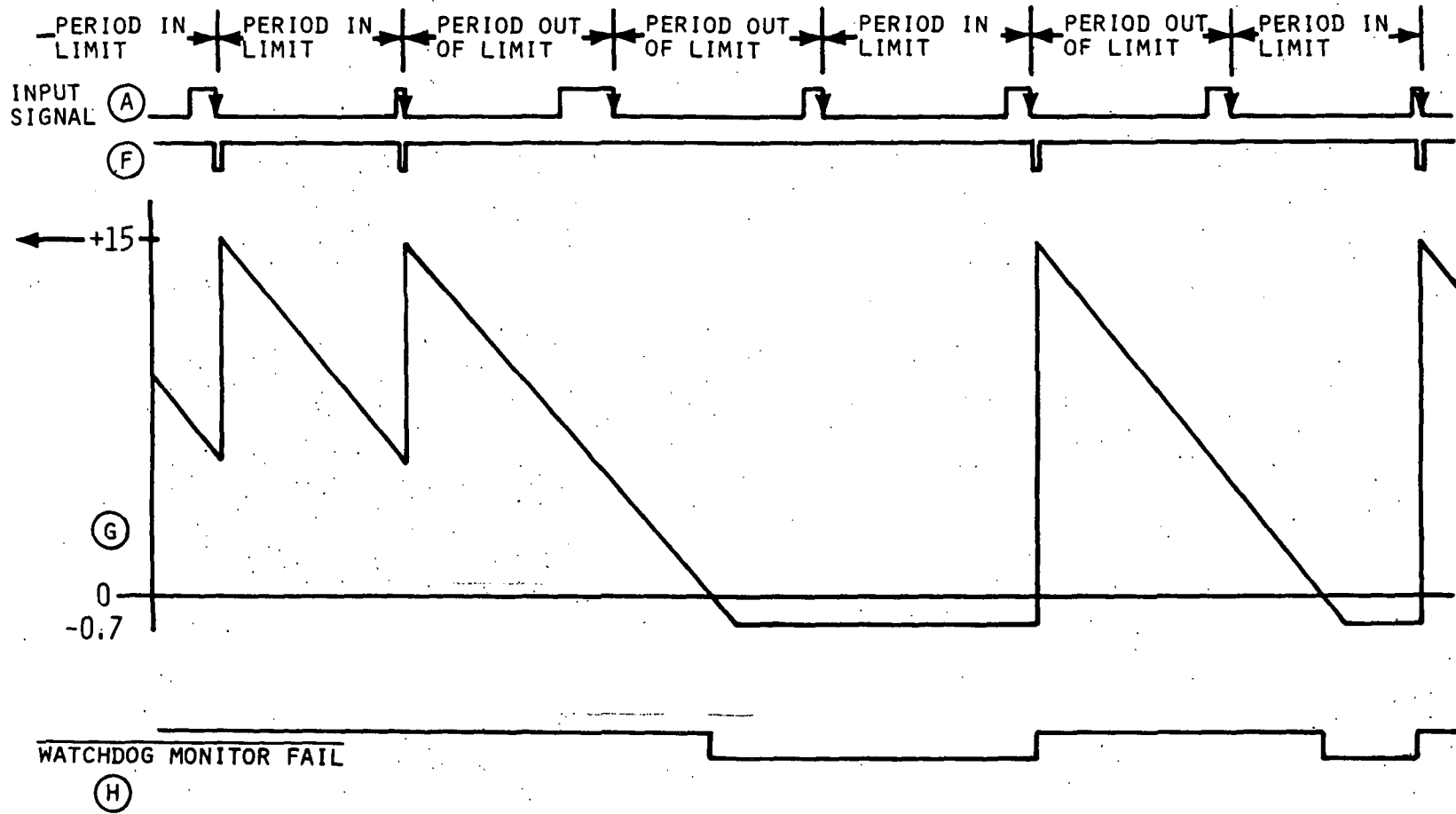


Figure 44.—Watchdog Monitor Timing Diagram

The ac-coupled monostable multivibrator is designed to generate a computer failure discrete at a specified time interval after an in-tolerance pulse (F) is received, unless it is "saved" from doing so by another in-tolerance pulse. In this manner the circuit is continually being "saved" from generating the computer failure discrete, and if anything happens to prevent pulse F from occurring, the discrete will be generated. This is fail-safe in that any failure mode of the input signal or the 4-MHz clock is covered.

5.3.3.5 Power Supply

The computer power supply must furnish +5 Vdc logic power and ± 15 Vdc analog/digital interface power. Figure 45 shows a block diagram of the supply. Primary input power is +28 Vdc with characteristics per MIL-STD-704A. EMI line filtering and transient suppression through a zener diode are provided. Conducted emissions are held down to the levels required by MIL-STD-461A.

The basic supply operates as a dc-to-dc converter with no intermediate conversion to ac and, consequently, no transformers. The +5 Vdc logic power is generated directly from +28 Vdc via a switching regulator. The +15 Vdc output is also generated directly from +28 Vdc but a simple linear series regulator is used. The -15 Vdc output is generated by a transformerless positive-dc-to-negative-dc converter. A switching regulator is used to chop +28 Vdc into a "swinging choke," which stores energy and then transfers it to the output capacitor via a "half-wave" rectifier.

A power status monitor generates a PS VALID signal for use as a priority interrupt to the CPU. This interrupt is what initiates automatic power-on processing.

In addition to the regulated dc power used by the computer, ac excitation for LVDT's in the servo loops is required. It is assumed that 400-Hz excitation is available in any application and that only a step-down transformer would be required. However, if OR'ed power redundancy is to be employed, then a dc-to-ac inverter operating from an OR'ed 28-Vdc bus would be required at some point in the system.

5.3.3.6 Built-In Test and Self-Monitoring Functions

Special considerations have been made in all areas of the ARCS hardware design to enhance the built-in test (BIT) and self-monitor capabilities. In particular, the following hardware self-monitor and built-in test functions have been designed into the ARCS computer unit.

Watchdog Monitor.—A watchdog monitor has been included to detect that class of fault for which the processor is no longer a logical element. The watchdog monitor must detect the falling edge of the LSC discrete at regular periods of $10\text{ ms} \pm 100\mu\text{s}$. The LSC is set and cleared under software control so the processor may use software-determined fault status to affect the state of the watchdog monitor. When the watchdog monitor trips, it disconnects all the servoactuators driven by that computer and provides a discrete to the processor.

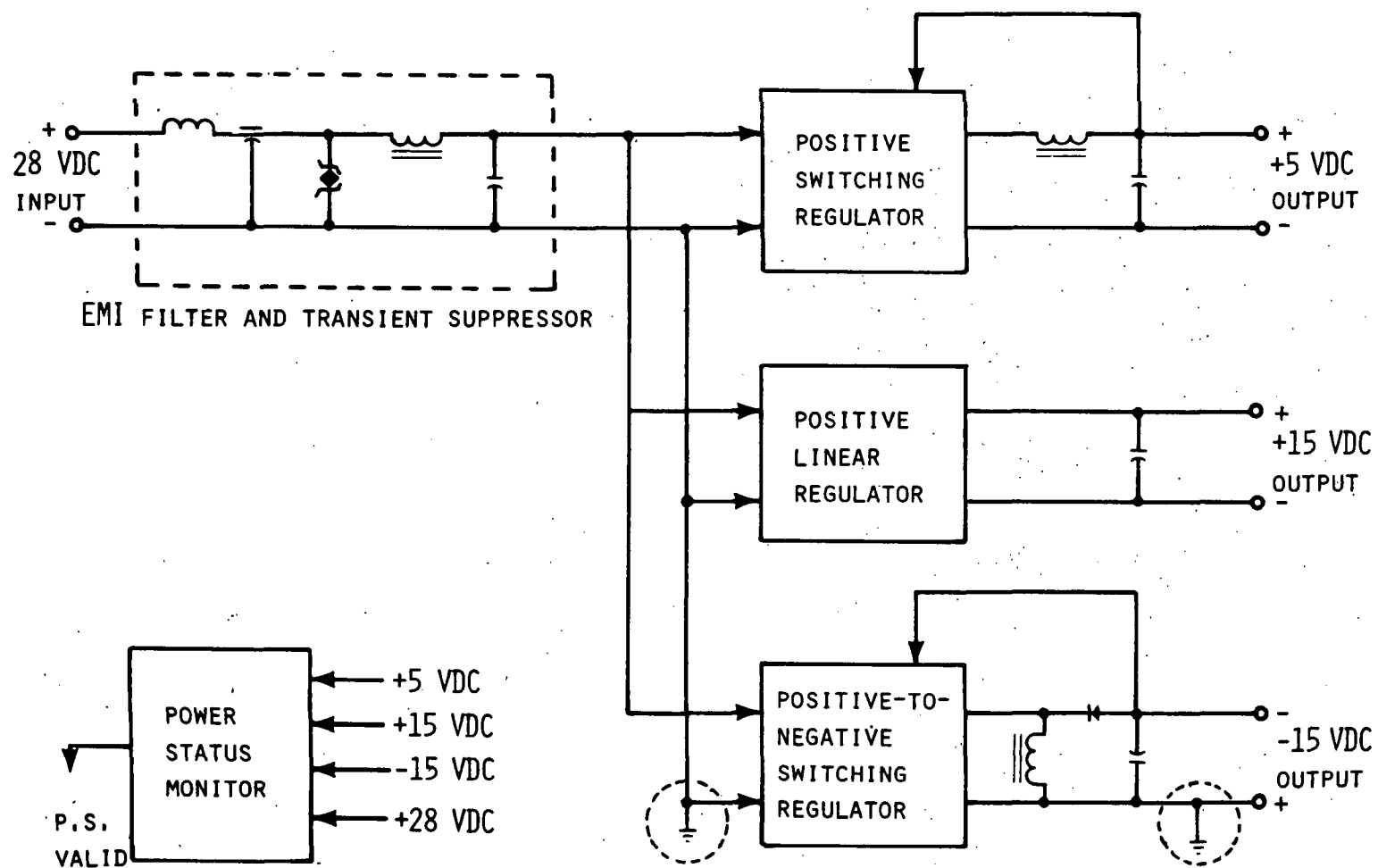


Figure 45.—Power Supply Block Diagram

Arithmetic Fault Detector.—The arithmetic fault detector monitors all arithmetic operations in the processor. It detects a fault whenever any of these arithmetic operations—addition, subtraction, multiplication, division, or arithmetic left shift—attempts to create a number that is outside the valid representation range of the computer (2's complement fractional representation). When a fault condition is detected, an interrupt is generated through the computer fault interrupt.

Power Supply Monitor.—A comparator monitor is used to detect out-of-tolerance conditions on all three secondary voltages, +5 and ± 15 V. An out-of-tolerance condition is used to disconnect all the servoactuators driven by that computer. A return to an in-tolerance condition produces a power-on interrupt. The processor is made cognizant of the actuator connection status through the state of the actuator shutoff valve discretes.

Scratch-pad Memory Parity Generator/Checker.—Odd parity is generated for all data being stored into the scratch-pad memory (RAM) and checked for all data being read out. If invalid parity is detected, an interrupt is generated through the computer fault interrupt. All subsequent activity is software determined. A parity generator inversion discrete is controllable from within the processor for use in testing the parity generator/monitor circuitry.

Asynchronous Digital Input/Output Validity.—In-line validity checking (parity plus bit framing where appropriate) is performed on all asynchronous digital input devices (inter-channel data link, system test panel data link, and SPBP). Invalid data detection results in an interrupt to the processor. Those receivers with a RAM capability will not store the invalid data in the RAM. The system test panel receiver will pass the data to the processor, with the validity bits in the upper byte of the word.

Discrete Input/Output Loop.—The discrete input/output module has been designed to provide loop testing capability within the module for all discrete outputs and inputs. The loops are activated by a self-test enable discrete that can be controlled by software.

Analog Input/Output Loop.—The analog input/output module has been designed to provide loop testing capability within the module for all analog inputs and outputs. The loops are activated by a self-test enable discrete that can be controlled by software.

Asynchronous Digital Input/Output Loop.—All asynchronous digital input/output modules have been designed to provide loop testing capability within the module through the transmitter and receiver(s). The loops are activated by a self-test enable discrete that can be controlled by software. Inverse parity generation within the respective transmitters is also controlled by a software-generated discrete.

Independent Servo Monitor.—An independent servo-loop monitor compares the two independent servo command signals for each servoactuator, which are summed in the servo-valve. This monitor detects failures of the multiplex D/A converter and sample and hold circuits in addition to failures within the servo loop. Upon detecting a failure, the independent monitor disconnects the servo involved and signals the processor through the state of the actuator shutoff valve (SOV) discrete.

5.3.4 SYSTEM TEST PANEL

The system test panel (STP) provides the control and display interface necessary for the flight and maintenance crews to perform various system tests. As indicated previously, the STP will communicate with each ARCS computer unit via a serial digital RS-232C data link. Figure 46 is a block diagram of the electronics installed within the panel to provide the required control, display, and interface.

The operation of the STP may be described two ways: in terms of basic hardware design functions or in terms of the system user functions that are determined by both hardware and software. Since the software aspects of system test are discussed in appendix B, this section will be concerned only with hardware design functions.

As shown in figure 46, the heart of the STP electronics is a single UART device. This LSI device provides the serial-to-parallel and parallel-to-serial conversions, as well as the signal formatting necessary for the RS-232C interface with the computers. The same serial data is transmitted simultaneously from the STP to each computer. One eight-bit word is transmitted each time a momentary switch is activated on the panel. The rotary mode control switches on the STP, shown in figure 47, will not cause any data transmissions when activated by themselves (the STP OFF position will, however, prevent transmissions). The content of the eight-bit word is determined by the encoding logic following the panel switches.

The STP receives data in eight-bit bytes from any one of the computers connected to it. Individual computer valid discretes are used to automatically select which computer's data is received by the panel. When all computers indicate valid operational status, then computer A data is the normal receiver selection. Two types of data are received: mode status annunciation data and alphanumeric display data. The first type is used to illuminate indicator lamps that confirm momentary switch activations or show status (e.g., the FAIL and ALERT indicators in fig. 47). The second type of data is an ASCII character string that provides a message for the alphanumeric display. The Burroughs' "self-scan" display (with memory) is the type used. The current Boeing requirement is for at least a 12-character display capability. Each character string message is held on the display until a new message is received from the computer.

Because the STP is intended to be installed within the flight deck area, it is required to operate without cooling air in a Category A1 environment per reference 3. Internally regulated dc power supplies provide panel logic power and display power. A dimmer-controlled 28-Vdc source powers all indicator lamps. The prime input power source for the STP is expected to be 28 Vdc. If necessary, redundant 28-Vdc sources may be OR'ed together to provide power to the panel in the event of power system failures.

The two basic functions of the STP are to *control* the test function and to *display* test results. The result of combining the airline operational needs with ARCS functional characteristics is represented by the panel layout shown in figure 47.

The STP will provide the following *control* features:

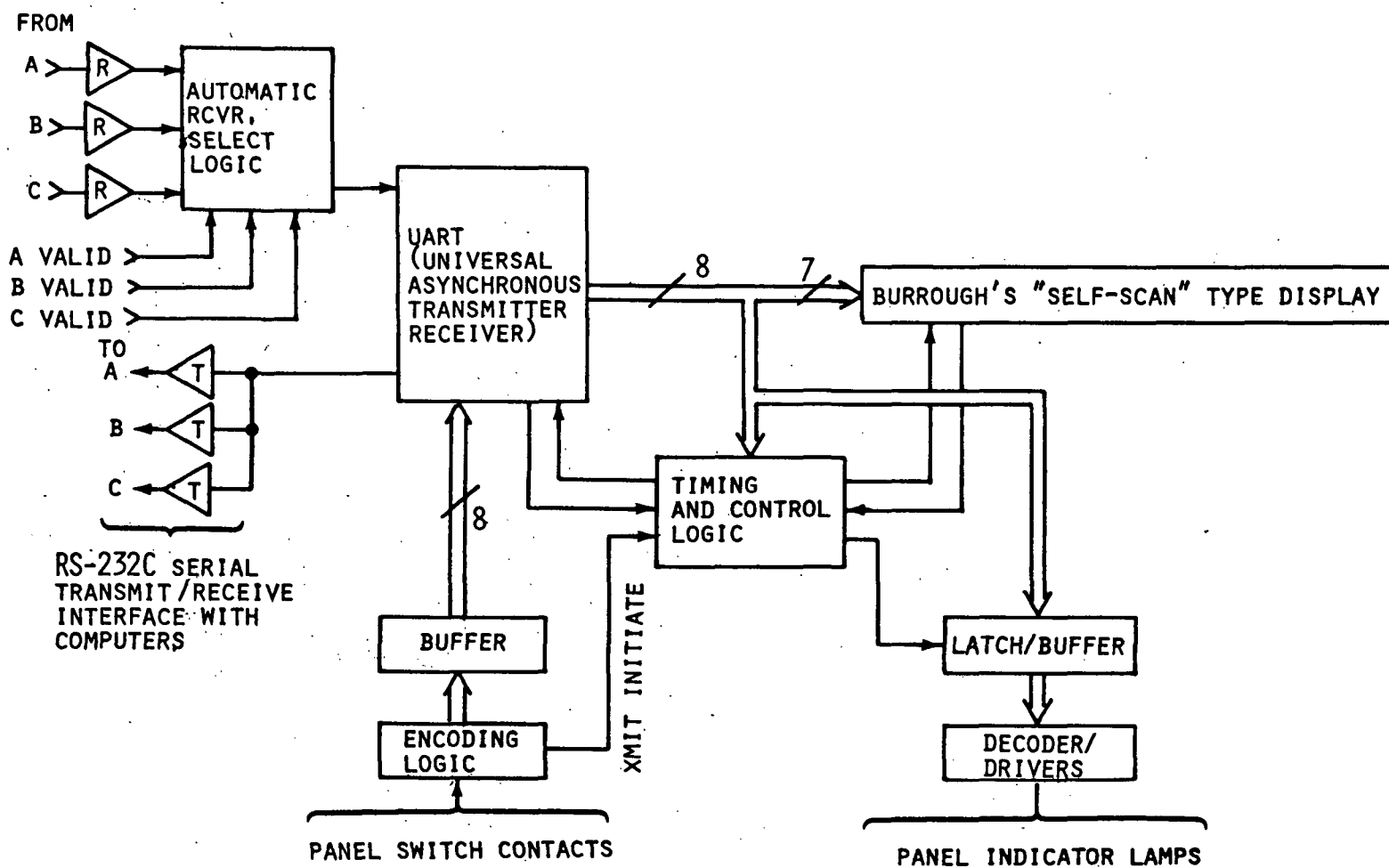


Figure 46.—System Test Panel Block Diagram

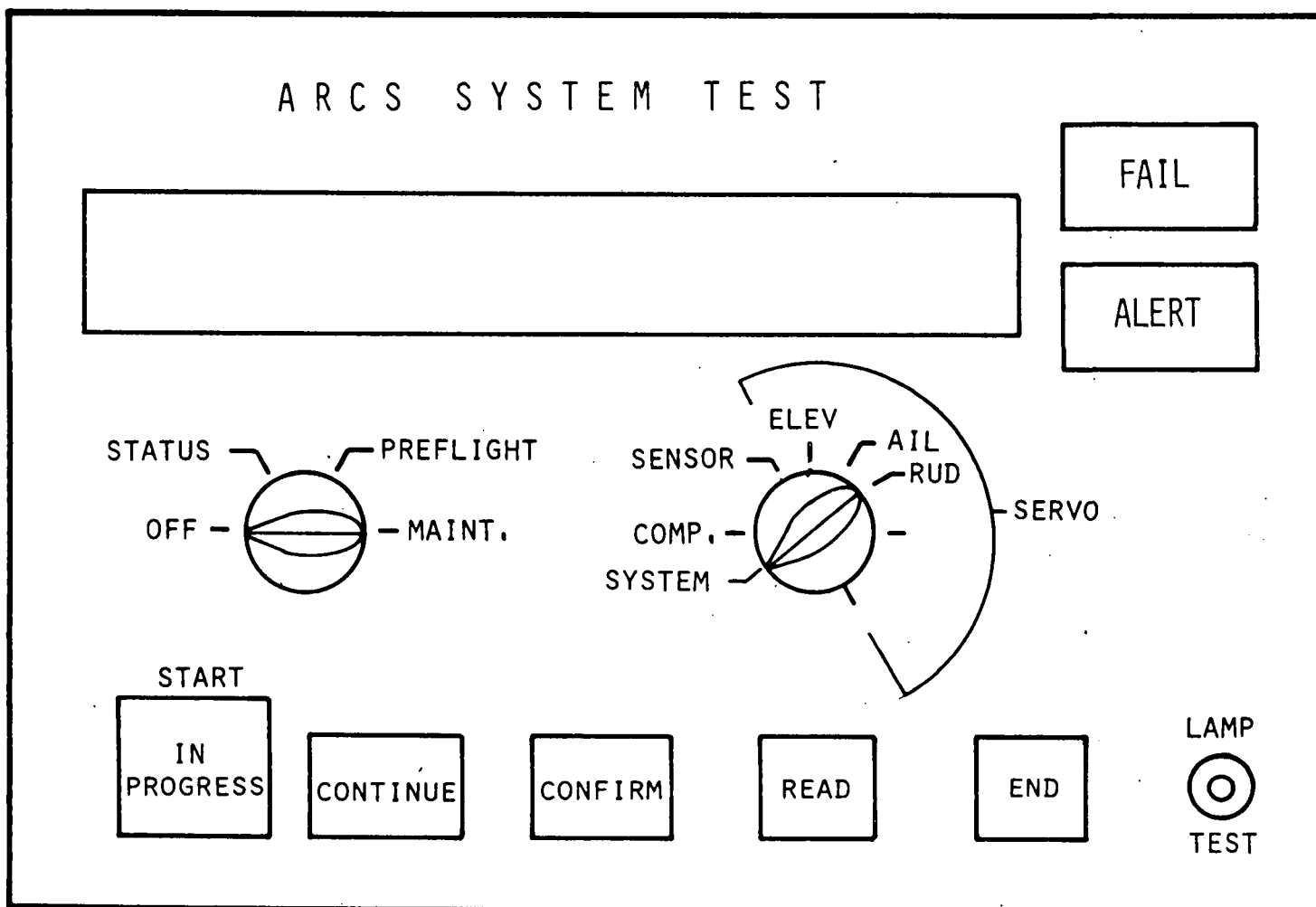


Figure 47.—ARCS System Test Panel Layout

START—A momentary action backlighted pushbutton for activation of the ground test function for preflight or maintenance purposes. Activation of the test function is acknowledged when the ARCS computers cause the button to be illuminated.

CONFIRM—A momentary action pushbutton by which the crew can confirm that the action required by the test in progress has been accomplished.

CONTINUE—A momentary action pushbutton by which the operator can cause the system to resume testing following a programmed stop. This might be the case when, for example, the computers are ready to begin servo testing but will not proceed until requested to do so.

END—A momentary action pushbutton which, when activated, will cause the ground test function to skip all remaining tests.

READ—A momentary action pushbutton that will cause the current failure status of the selected test to be displayed. Repeated activation of this function will cause sequential messages from computer memory to be displayed.

Test mode selector—A manual rotary switch with four positions: OFF, STATUS, PREFLIGHT, and MAINT. This switch position causes the ARCS computers to respond with the corresponding system test function, e.g., STATUS will cause a display of the current operational status of the system, PREFLIGHT will cause a preflight test to be initiated when the start button is depressed, and MAINT will select a full ground test.

Test function selector—An eight-position rotary switch that allows the operator the flexibility of selecting either a test of the entire system or some particular subset of the total system. Selectable functions include SYSTEM, SENSORS, COMPUTERS, or any of the SERVO-actuators.

Lamp test—A lamp test button which, when activated, will verify all indicator lamps and display elements.

The STP will provide the following *display* features:

FAIL—A red FAIL light that will illuminate when a failure has been detected which is applicable to the currently selected mode or which results in a loss of dispatch capability of the airplane.

ALERT—A yellow ALERT light that will illuminate when a failure condition has been detected which does not affect the current mode of operation or the dispatch capability of the system.

Readout—A test readout capable of displaying a minimum of 10 alphanumeric characters. This display will be used to enunciate LRU-level failure information, system status or operational capability, operator action requirements, or any other information or cues that require a text-type readout.

6.0 ARCS DESIGN ANALYSIS

The purpose and scope of the analytical work performed to synthesize an ARCS design concept and establish its fault-tolerant characteristics is illustrated by figure 48. The illustration shows the relationships of activities involved in a complete development of a fault-tolerant system from design to actual system (hardware and software) testing for design verification. Activities bounded by solid frames were within the scope of the ARCS program. The design synthesis resulting in the system concept described in section 5 is summarized in appendix D.

The first part of this section (sec. 6.1) deals with the development of analytical tools for measuring or assessing the fault tolerance of a redundant computer system, a primary activity within the ARCS program. This part describes the approaches and results of the fault analysis, reliability model synthesis, and probability projections performed for the ARCS and the baseline WWCS.

Closely related to the specification of fault-tolerance requirements, and to the analysis performed during the design phase, is the verification of the fault-tolerant performance of the fully mechanized system; refer again to figure 48. During the ARCS study, General Electric, working closely with Boeing, developed a new approach for measuring coverage. This approach is based on a failure analysis of a randomly selected set of failure modes extracted from the entire failure mode population. The main theme of this method is discussed in section 6.1.

The second part of this section (sec. 6.2) is an assessment of the cost effectiveness of applying ARCS technology, carried out in two parts: an analysis of airline cost-of-ownership for an ARCS maintained in a Category III operational status and an analysis of the cost effect of providing an integrated system test function.

6.1 FAULT-TOLERANCE ANALYSIS

The delineation of the ARCS fault-tolerance analysis is organized into three parts, as illustrated in figure 49. The fault analysis is a prerequisite for the synthesis of a reliability model, and the reliability model is used to derive success probability projections for a given design with a given set of input assumptions.

Section 6.1.1 discusses the approach used, and the functional simulation applied, to establish confidence that the ARCS design will indeed provide fault-tolerant performance. The reliability modeling description, introducing the computer programs used to compile the WWCS and ARCS reliability estimations, and the results of the WWCS and ARCS reliability estimations are presented in section 6.1.2.

6.1.1 FAULT ANALYSIS

In general, system fault analysis has two primary purposes: to aid in developing a workable system concept during the design phase and to validate the system design once the design is complete. Only the first applies in the ARCS study, where the particular objectives of

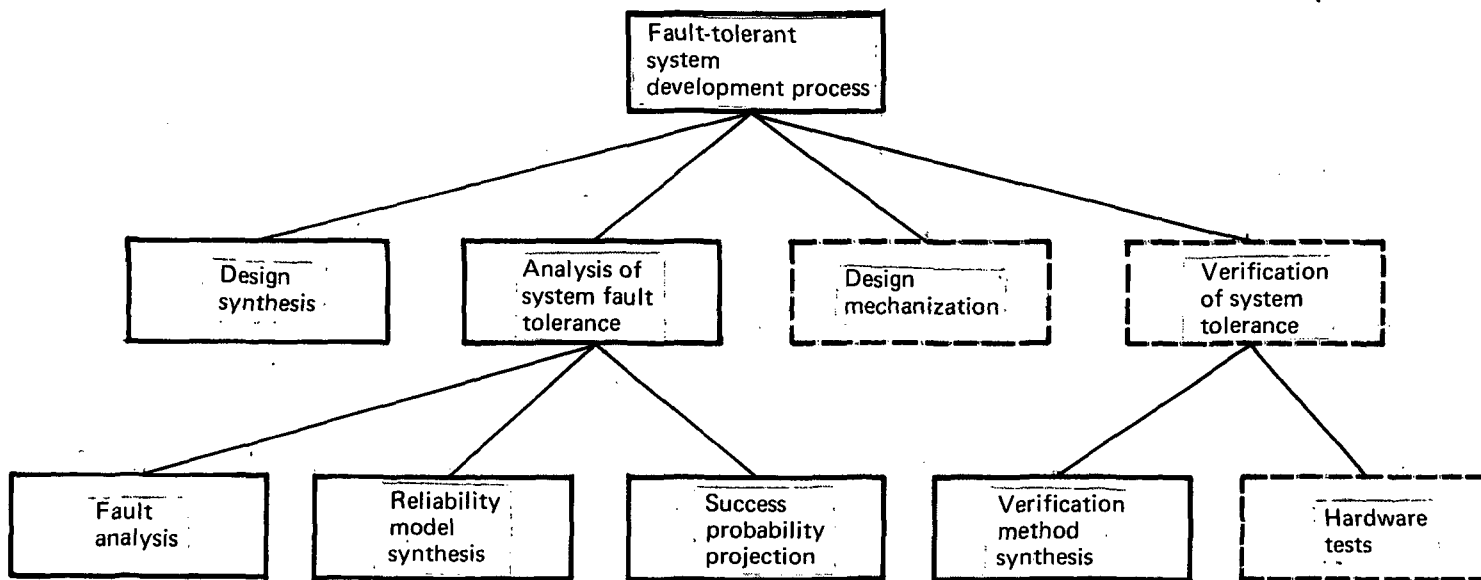


Figure 48.—ARCS Synthesis/Analysis Scope

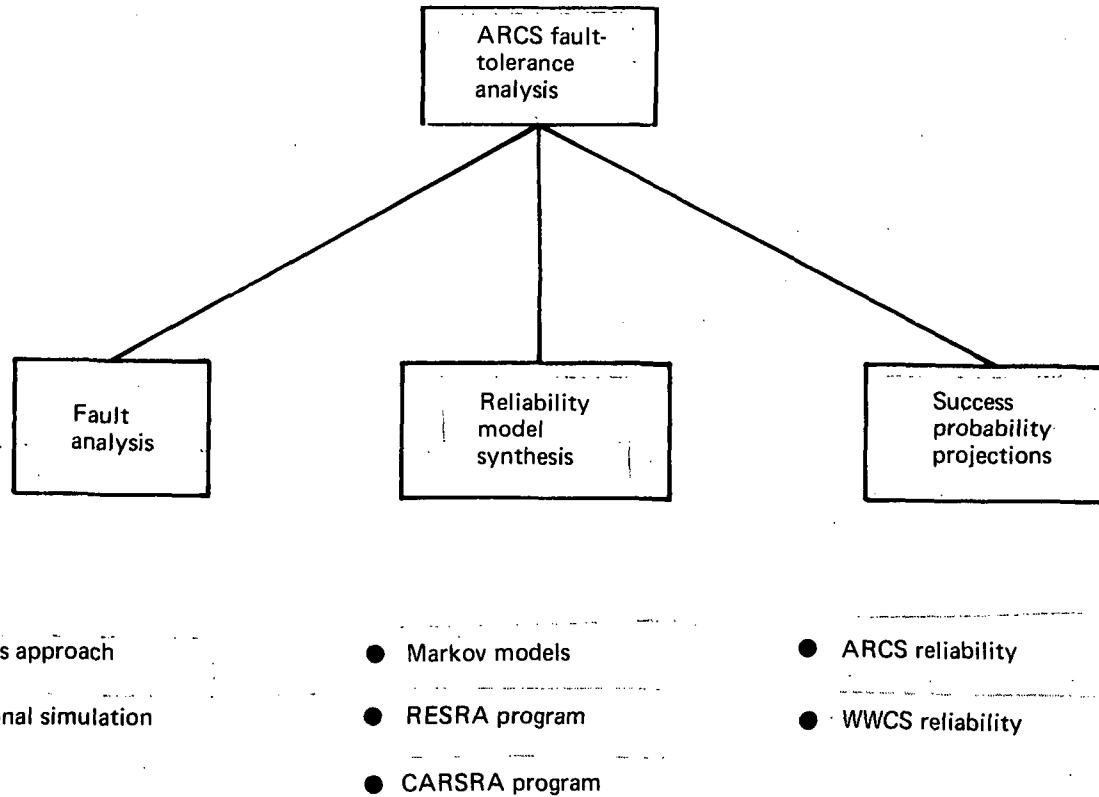


Figure 49.—ARCS/WWCS Fault-Tolerance Analysis Overview

the fault analysis task were to (1) develop a system concept that guarantees first-failure survivability and (2) evaluate the viability of the developed reconfiguration processes.

The ARCS fault tolerance is founded on the use of three or more autonomous redundant computers exchanging data via digital data buses, each providing a mechanical output into a mechanical voting mechanism having complete fault tolerance for first failure. The technology to realize the complete isolation from nonintentional cross-channel effects, and to realize a sufficiently reliable mechanical voter, was judged to be state of the art.

With the above assumptions, the following breakdown of detectable fault classes can be made based on the relationship of the fault to the different monitor functions presented in the ARCS design concept of section 5.

<u>Fault class</u>	<u>Associated monitor</u>
Any failure causing a processor not to complete tasks on schedule	Watchdog monitor
Any failure causing a processor to produce dissimilar results	Output monitor
Any fault upstream of the virtual SSFD voting node	SSFD
Any fault downstream of the computer output monitor node	Servo monitor
Electrical power loss	(Trivial)
Any failure causing corruption of cross-channel data between two computer units	Cross-channel monitor

For the level of design detail that the scope of the ARCS program allows, a further breakdown of fault classes is not meaningful. The purpose of the ARCS fault-tolerance analysis was therefore to validate that the design concept will handle the classes of faults listed above through the range of triplex, duplex, and simplex states, as well as transient fault recoveries in each of these redundancy states.

For that purpose, the ARCS concept was analyzed on paper and was implemented into a functional simulation—the Reconfigurable Computer System Simulation (RCSS)—representing three synchronized but autonomous computers processing the algorithms required to perform all the ARCS processes that are significant to the fault-tolerant capability of the system. The simulation was used as a tool in the fault analysis to help evaluate the reconfiguration processes. These processes, though each conceptually understandable, are difficult to validate using only a paper analysis.

6.1.1.1 Paper Analysis

Several levels of paper analysis can be performed on a design concept such as the ARCS. The synchronization process was examined as a part of the design analysis (see app. D) to ensure that the selected algorithms met the basic requirements. The initial synchronization was examined from a system point of view (see sec. 5.1) to ensure that the selected algorithms accomplished the system requirements as prescribed in section 4.2.

The recovery strategy of table 3 was analyzed and used to specify the sequential machine of figure 50. The 16 states of figure 50 represent the acceptance settings of the transient and permanent failure flags defined in table 3. In this figure, a state such as $q_1(LX\bar{Y})$ indicates that the transient failure flag of Y is set. (See table 3, third channel synchronized before sensor settling.) A state such as $q_4(LX)$, Y not present, indicates that the Y permanent flag is set. (See table 3, two-channel operation after sensor settling.)

If the local computer determines that it is simplex, it will start in state $q_{15}(L)$. If it determines that another computer is already operating (i.e., duplex recovery), it will recover to it and start in states $q_5(\bar{L}X)$ or $q_9(\bar{L}Y)$, depending on whether the other computer is to its left or right. If it determines that two other computers are operating, it will recover to one of them. If no transient failure flag is set, the local computer begins operation in $q_2(\bar{L}XY)$. If a transient failure flag is set, it will begin operation in $q_6(X\bar{Y})$ or $q_8(\bar{X}Y)$. When the flag is cleared, it will transition to $q_7(XY)$ from which recovery to $q_2(\bar{L}XY)$ can then proceed. Had the fault associated with q_6 or q_8 been declared permanent, the local computer would have transitioned to $q_{13}(X)$ or $q_{14}(Y)$, from which recovery to $q_5(\bar{L}X)$ or $q_9(\bar{L}Y)$ could proceed.)

In figures 51, 52, and 53, the table at the top of each figure gives the sequence of states through which the software would transition from each computer's point of view. The first column specifies the local (L), left (X), and right (Y) channels.

Figure 51 shows the power-on sequence for the three computers. Computer No. 1 comes on, notes that it is alone, and proceeds to operate in simplex. Computer No. 2 comes on, notes that No. 1 is operating and recovers to it. Computer No. 3 comes on, notes that Nos. 1 and 2 are operating and that No. 2's do-not-use flag is set, and waits for the flag to clear before attempting recovery. The key point here is that as long as a transient failure flag is set, the third computer will not attempt to recover. If computer No. 2 fails to recover, it will be permanently faulted and No. 3 will attempt to recover, as shown in figure 52.


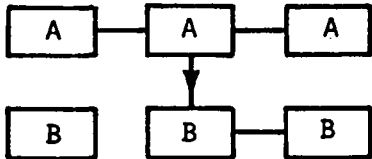
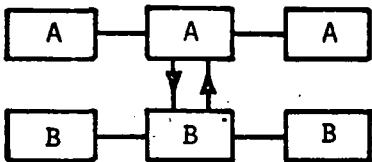
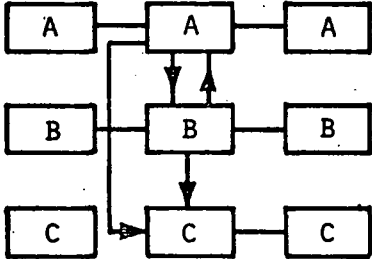
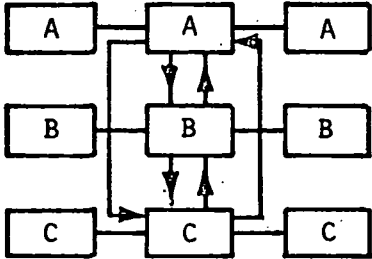
Figure 53 shows the case where No. 2 recovers and No. 3 starts recovery but No. 2 detects a fault. In this case, No. 3 is faulted. However, since No. 3 will assess that it is fault free, it will get another chance at recovery as soon as the fault assessment in No. 2 is resolved.

Figure 54 shows the case where the system function is triplex and a fault is detected and declared permanent.

6.1.1.2 Simulation Results

The ARCS simulation using the general simulation program RCSS (Redundant Computer

Table 3.—Recovery Strategy

OPERATION	TRANSIENT FAILURE FLAGS	PERMANENT FAILURE FLAGS
SIMPLEX SENSORS COMPUTER SERVOS 	A B C 0 0 0	A B C 0 1 1
AFTER SECOND CHANNEL SYNCHRONIZED BEFORE SENSOR SETTling 	0 1 0	0 0 1
AFTER SENSOR SETTling 	0 0 0	0 0 1
THIRD CHANNEL SYNCHRONIZED BEFORE SENSOR SETTling 	0 0 1	0 0 0
AFTER SENSOR SETTling 	0 0 0	0 0 0

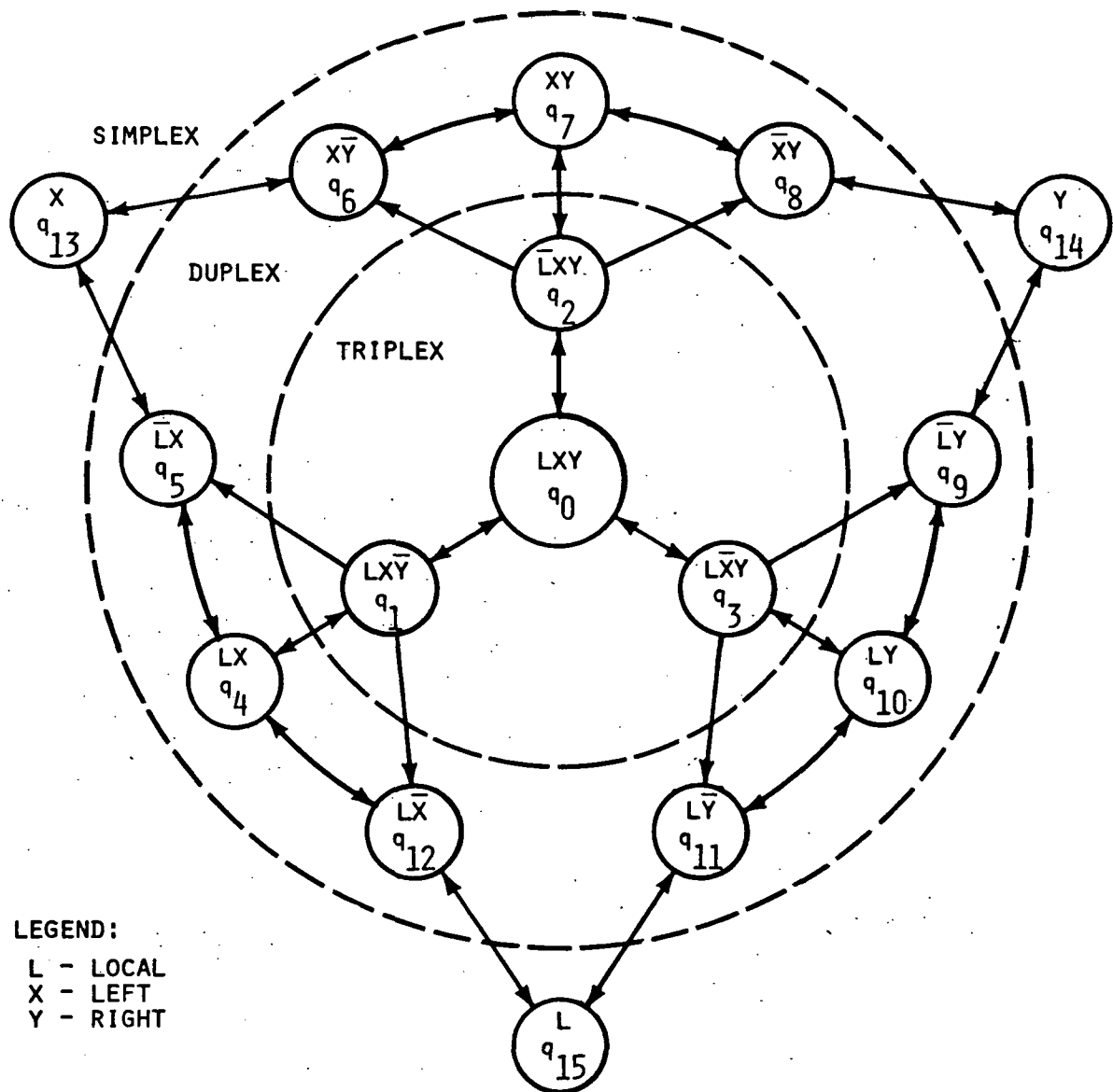
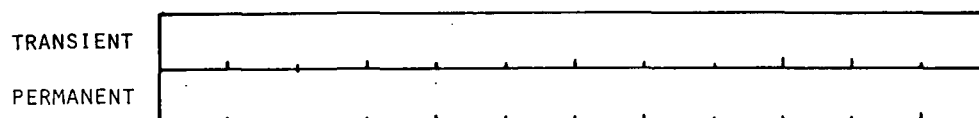


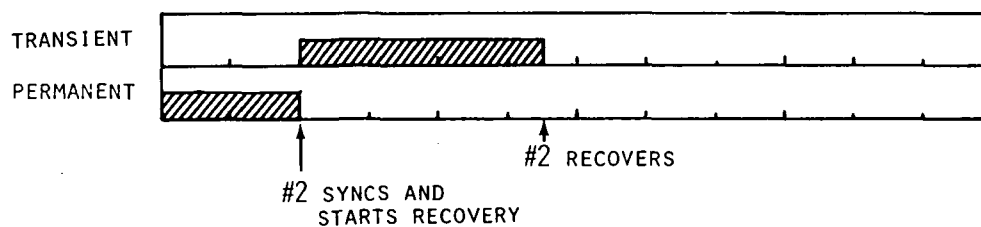
Figure 50.—Local Computer's Assessment of a System Function

frame LXY	1	2	3	4	5	6	7	8	9	10	11	12
132	q ₁₅	q ₁₅	q ₁₁	q ₁₁	q ₁₁	q ₁₁ / q ₁₀	q ₃	q ₃	q ₃	q ₀	q ₀	q ₀
213	—	—	q ₅	q ₅	q ₅	q ₅ / q ₄	q ₁	q ₁	q ₁	q ₀	q ₀	q ₀
321	—	—	—	q ₈	q ₈	q ₈ / q ₇	q ₂	q ₂	q ₂	q ₀	q ₀	q ₀

(#1)



(#2)



(#3)

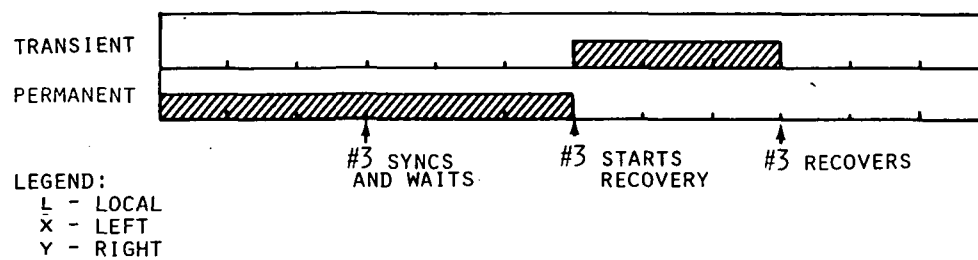
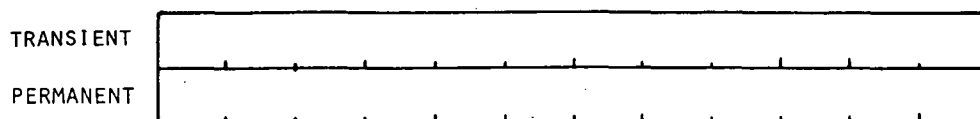


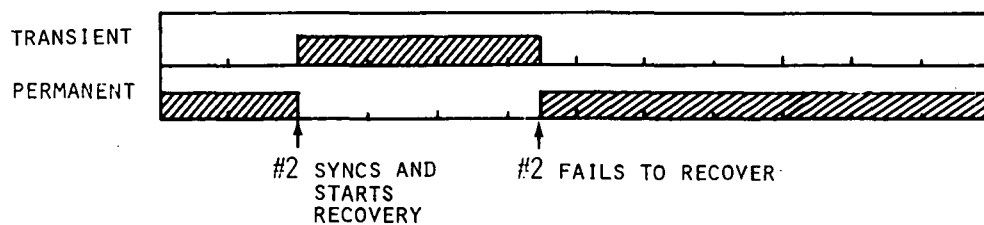
Figure 51.—Power-On/Watchdog Monitor Trip Recovery as Processed by an Operating Computer (Triplex Operation Attained)

LXY \ frame	1	2	3	4	5	6	7	8	9	10	11	12
132	q ₁₅	q ₁₅	q ₁₁	q ₁₁	q ₁₁	q ₁₁ / q ₁₅	q ₁₂	q ₁₂	q ₁₂	q ₄	q ₄	q ₄
213	—	—	q ₅	q ₅	q ₅	q ₅ / q ₁₃	q ₆	q ₆	q ₆	q ₇	q ₇	q ₇
321	—	—	—	q ₈	q ₈	q ₈ / q ₁₄	q ₉	q ₉	q ₉	q ₁₀	q ₁₀	q ₁₀

(#1)



(#2)



(#3)

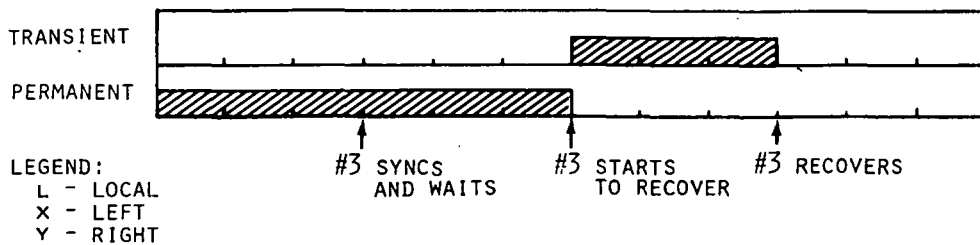


Figure 52.—Power-On and Watchdog Monitor Trip Recovery as Processed by an Operating Computer (Duplex Operation Attained)

frame LXY	1	2	3	4	5	6	7	8	9	10	11	12
132	q ₁₅	q ₁₅	q ₁₁	q ₁₁ / q ₁₀	q ₃	q ₃ / q ₁₁	q ₁₁	q ₁₁	q ₁₁	q ₁₁	q ₁₅	q ₁₅
213	—	—	q ₅	q ₅ / q ₄	q ₁	q ₁ / q ₅	q ₅	q ₅	q ₅	q ₅	q ₁₃	q ₁₃
321	—	—	—	q ₈ / q ₇	q ₂	q ₂ / q ₈	q ₈	q ₈	q ₈	q ₈	q ₈	q ₈

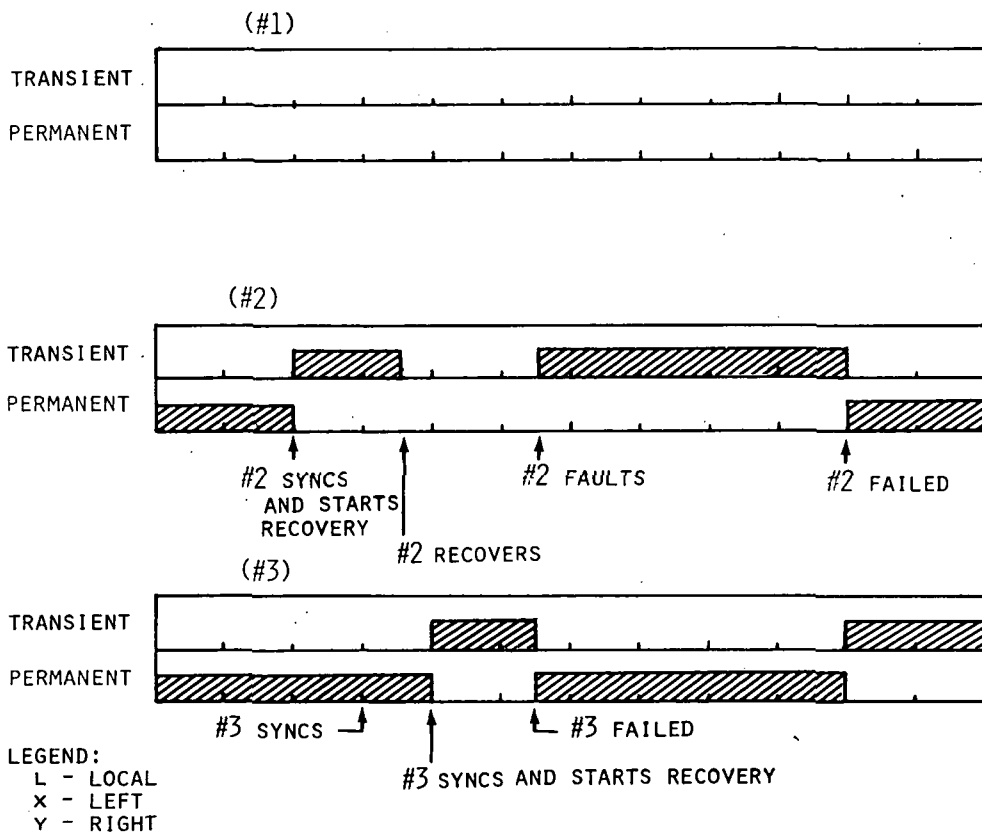
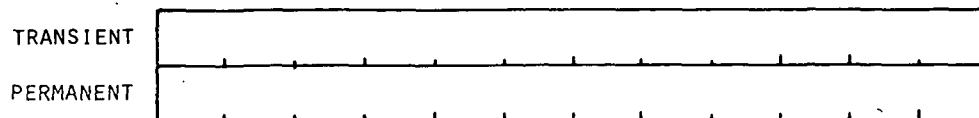


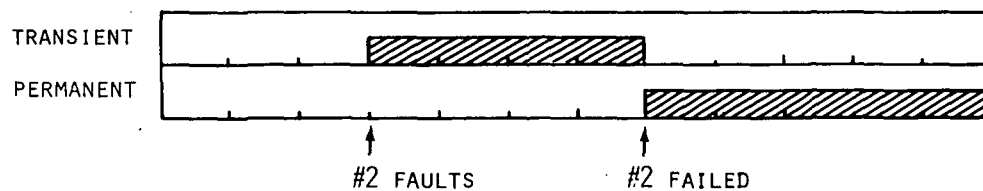
Figure 53.—Power-On/Watchdog Monitor Trip Recovery as Processed by an Operating Computer (Simplex Operation Attained)

frame LXY	1	2	3	4	5	6	7	8	9	10	11	12
132	q ₀	q ₀	q ₀	q ₁	q ₁	q ₁	q ₁	q ₄	q ₄	q ₄	q ₄	q ₄
213	q ₀	q ₀	q ₀	q ₂	q ₂	q ₂	q ₂	q ₇	q ₇	q ₇	q ₇	q ₇
321	q ₀	q ₀	q ₀	q ₃	q ₃	q ₃	q ₃	q ₁₀	q ₁₀	q ₁₀	q ₁₀	q ₁₀

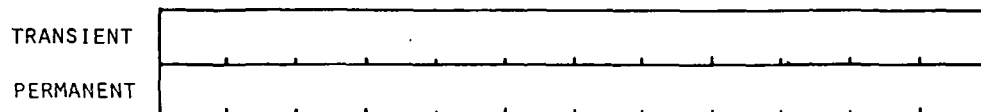
(#1)



(#2)



(#3)



LEGEND:
 L - LOCAL
 X - LEFT
 Y - RIGHT

Figure 54.—Power-On/Watchdog Monitor Trip as Processed by
 an Operating Computer (Triplex-to-Duplex Degradation)

System Simulation) was used to (1) explore various recovery concepts and (2) verify reconfiguration design completeness.

The simulation treats parallel real-time processing for the redundant channel configuration sequentially, subprocess by subprocess. After the data resulting from the real-time-referenced simulation processing of an ARCS subprocess has been sequentially performed and exchanged between the three simulated processors, processing for the next subprocess between data exchange points is performed, and so on until the frame processing is completed. Appendix E shows the use of the simulation to demonstrate the power-on/power-fault reconfiguration process and illustrates how the simulation was used to help verify the ARCS design concept.

In the area of recovery, two concepts were explored. The first was that of each system function handling complete recovery for that function. This concept gives functions such as the SSFD access to the channel synchronization status. Although workable, this concept was found to contribute to extremely complex data management.

The second concept—a hierarchy of recovery—allows the consolidation of recovery and redundancy degradation into one function, based on synchronization status and cross-channel monitor results. If synchronization is not attained or the cross-channel data link transmits erroneous data, the offending channel's permanent flags will be set by the recovery/redundancy degradation process.

The hierarchy of recovery concept was selected because it reduces data management and should ultimately reduce execution time by terminating recovery at the first level that recovery is impossible. The savings in data management would be realized since lower level processes, such as the SSFD, would not be required to have access to the synchronization status. This should reduce the complexity of the data interface. Execution time should also be reduced, since the subroutine calling sequence would be simplified. However, the importance of any reduction can be assessed only after software implementation.

Design completeness was verified by simulating a number of cases and observing the result. Due to cost and time, this verification was not exhaustive, but it did show the benefit of such processing by pointing out both design and implementation errors.

The magnitude of the reconfiguration problem was brought out mainly by the shear volume of data that must be recorded to analyze results. Appendix E describes several power-on and power-fault simulations.

6.1.2 RELIABILITY ANALYSIS

The ARCS reliability analysis had two main objectives: (1) to survey available reliability assessment tools and establish a reliability assessment technique applicable to the analysis of a redundant reconfigurable system and (2) to exercise this technique to predict reliability numbers for the WWCS and ARCS baseline configurations and for several ARCS trade study configurations.

Section 6.1.2.1 discusses some currently available reliability estimation methods and gives an account of the considerations that led to the development of a new reliability assessment

tool, CARSRA (Computer Aided Redundant System Reliability Analysis). CARSRA is a FORTRAN program specifically developed during the ARCS program. A brief account of this program is given below and a detailed description may be found in appendix F, which also contains a user's guide.

Section 6.1.2.2 describes the essence of the new method developed to assess coverage parameters. Details of this method are contained in appendix G.

Section 6.1.2.3 presents the main reliability results and conclusions from the ARCS/WWCS reliability study. A detailed account of the background data needed for the analyses is given in appendix H.

Section 6.1.2.4 summarizes conclusions and observations from the ARCS reliability analysis.

6.1.2.1 The Reliability Assessment Technique

The ARCS exhibits several unique properties that complicate the reliability analysis, one of which is the ability to degrade from triplex to duplex redundancy following a first failure, and from duplex to simplex upon the majority of all second failures. The conditional probability that the system survives given a second failure is denoted "the second-failure coverage" for the system. This parameter is of great significance in the reliability assessment and should be incorporated into any viable reliability model.

Another property of ARCS is its ability to survive most transient faults. Modeling of transient faults further complicates the reliability analysis. Although nonsurvivable transients conceivably could be modeled by increasing the permanent equipment failure rates, this increase would have to depend on the operational redundancy level since the system is more likely to survive transients at triplex redundancy than when operating in duplex.

A special requirement for the ARCS reliability study was the need to evaluate system functional readiness as well as system failure probability. Functional readiness is the probability of operating at some prescribed system redundancy state (defined as the functional readiness criterion) at a particular time. This parameter is of interest for fault-tolerant systems that are capable of sustaining module failure(s) without losing the availability of the system function. For the ARCS, functional readiness is of interest in the context of permitting Cat III or STOL landings with certain parts of the flight control system failed. This will have the benefit of substantially increasing the Cat III and STOL function availability.

Survey of Available Reliability Assessment Tools and Approaches.—There exists a rich proliferation of reliability assessment tools, most of them in the form of digital computer programs. Table 4 displays analysis methods used by 10 different reliability estimation programs, 6 of which are currently in use at Boeing. Descriptions of the CARE, CARE2, CAST, and TASRA programs were obtained from NASA/LRC.

Table 5 shows different program features. Of the methods shown in table 4, the Markov model and simulation are the only techniques capable of handling the special requirements of the ARCS analysis.

Table 4.—Surveyed Reliability Programs and Associated Analysis Methods

Program name	Method				Standard configurations	Markov model	Simulation
	Event tabulation	Binomial expansion	Boolean algebra	Conditional probabilities			
ARMM (B)	X			X			
BINO (B)		X					
CARE					X		
CARE2					X		
CAST						X	X
CEPFRA (B)						X	
COBRA (B)			X				
CRAM (B)	X						
SIM (B)							X
TASRA	X						

(B) = programs used by Boeing

Table 5.—Reliability Program Features

Program name	Feature					
	Stage dependencies	Active redundancy	Standby redundancy	Permanent fault coverage	Transient fault recovery	Multiple dissimilar failure modes
ARMM (B)	X	X	X			X
BINO (B)		X				
CARE		X	X	(X)		
CARE2		X	X	X	X	
CAST	(X)	X		X	X	(X)
CEPFRA (B)	(X)	X	X	X	X	X
COBRA (B)		X				
CRAM (B)		X				
SIM (B)		X				
TASRA	X	X				X

(X) = limited ability

(B) = programs used by Boeing

The CAST program was developed by Ultrasystems under a contract from NASA/LRC. It is a two-step approach that uses a Monte Carlo simulation to estimate parameters in a Markov model describing the system. This Markov model is in turn used to evaluate system reliability. The approach presumes that the system is simple enough to permit modeling without having to deal with an exorbitant number of Markov states.

CEPFRA is a Boeing program based on the Markov approach. Like CAST, it presumes a relatively small number of Markov states (< 35). Since several hundred states would be needed to model the ARCS by a single Markov model, the applicability of the CAST and the CEPFRA programs to the ARCS reliability assessment task is limited.

ARCS-Developed Analysis Tools.—Because of the limitations of currently available computerized reliability analysis tools, the decision was made to develop a new tool—CARSRA (Computer Aided Redundant System Reliability Analysis).

CARSRA is a general-purpose reliability analysis program that handles modular-redundant reconfigurable systems taking into account such factors as coverage and transient faults. It evaluates functional readiness and system failure probabilities for two different failure modes, using a unique approach that combines Markov modeling with dependency tabulation. As a result, the dimensional problem of the Markov model is overcome by system partitioning into stages, or sets of redundant identical modules, so that each stage may be modeled by a dedicated Markov model of low order. The details of this approach as well as a guide to the use of the program may be found in appendix F.

In the process of developing CARSRA, a simplified version was written called RESRA (Redundant System Reliability Analysis). RESRA uses the same system-partitioning approach as CARSRA, with the difference that the stage reliabilities are expressed by closed-form analytical expressions rather than via a Markov model. In addition, RESRA does not compute functional readiness. RESRA requires less input information and is therefore easier to use than CARSRA. CARSRA was used to generate all the reliability data presented in the following.

For the purpose of the reliability analysis, the system is partitioned into stages and modules, where a module is defined as a set of elements performing a specified function. Each stage is modeled by a Markov model describing the different redundancy states of the stage. The two last states (assigned the highest numbers) in the Markov model are stage failure states. The next to the last state corresponds to a detected failure and the last state to an undetected failure. Up to nine states in the Markov model are permitted for each state. An example of a stage Markov model is given in figure 55.

Depending on the way in which the various stages have been defined, some stages will generally have the property that failure of one module in the stage will cause loss of function of a module in one or several other stages. This introduces a statistical dependency between the various stages that has to be taken into account in the analysis.

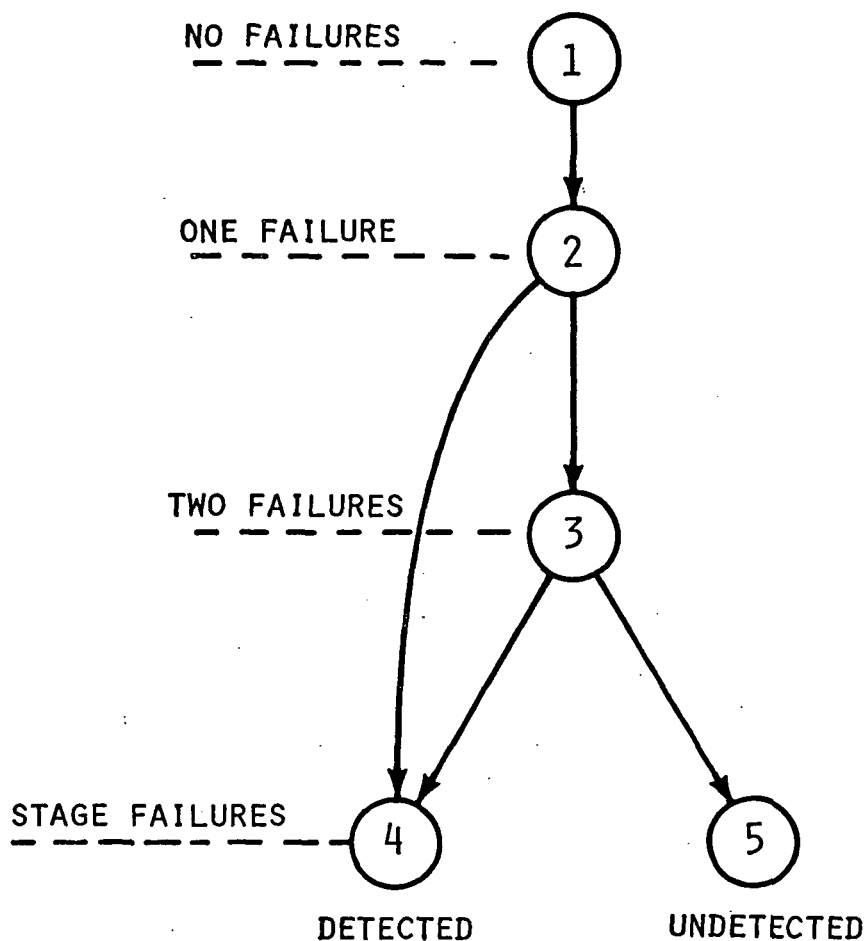


Figure 55.—Example of Markov Model of a Triplex Stage

The system dependency structure may conveniently be described by a dependency tree diagram, an example of which is shown in figure 56. Dependencies between the various stages are indicated by the lines connecting the stages, and the "direction" of a dependency is indicated by an arrowhead. The dependency tree may be used to establish the dependency table by which the system dependency structure is specified for CARSRA. The numbers to the right of the stage blocks of figure 56 refer to the modular redundancy level of the stage.

Functional readiness, $FR(t)$, is defined as being the probability that a certain prescribed function is available after some time, t , of system operation. Thus, the FR is equal to unity at time zero and decreases toward zero for long exposure times. System functional readiness for selected module failure patterns will be computer by CARSRA. If, for

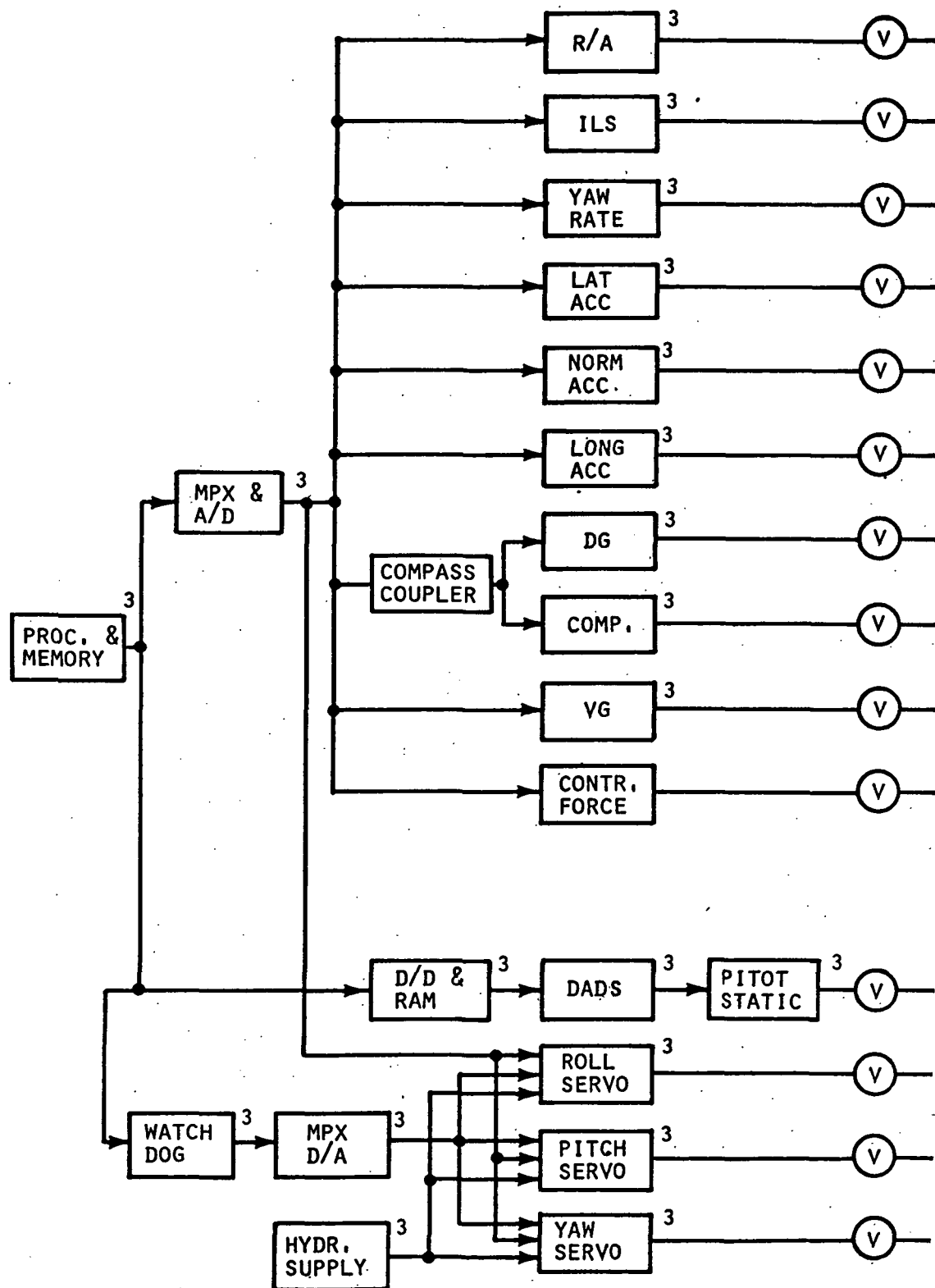


Figure 56.—Flight Control System Dependency Tree

example, n different module failure patterns are specified, the program will compute the probability of having each of these failure patterns at a time t : $FR_i(t)$, $i=1, 2, 3 \dots n$. The sum of these probabilities yields the functional readiness, $FR(t)$, corresponding to the *functional readiness criterion* consisting of the n specified module failure conditions:

$$FR(t) = \sum_{i=1}^n FR_i(t)$$

In this context it is also of interest to assess the probability of system failure given a certain functional readiness criterion. CARSRA computes this quantity. The probability of system failure at a certain time T from the beginning of the critical mission phase (for example, from alert height in a Cat III landing) is computed given each of the n module failure configurations in the functional readiness criterion. Let these failure probabilities be $FP_i(T)$. The system failure probability $FP(t, T)$ given a prescribed functional readiness criterion is then evaluated by:

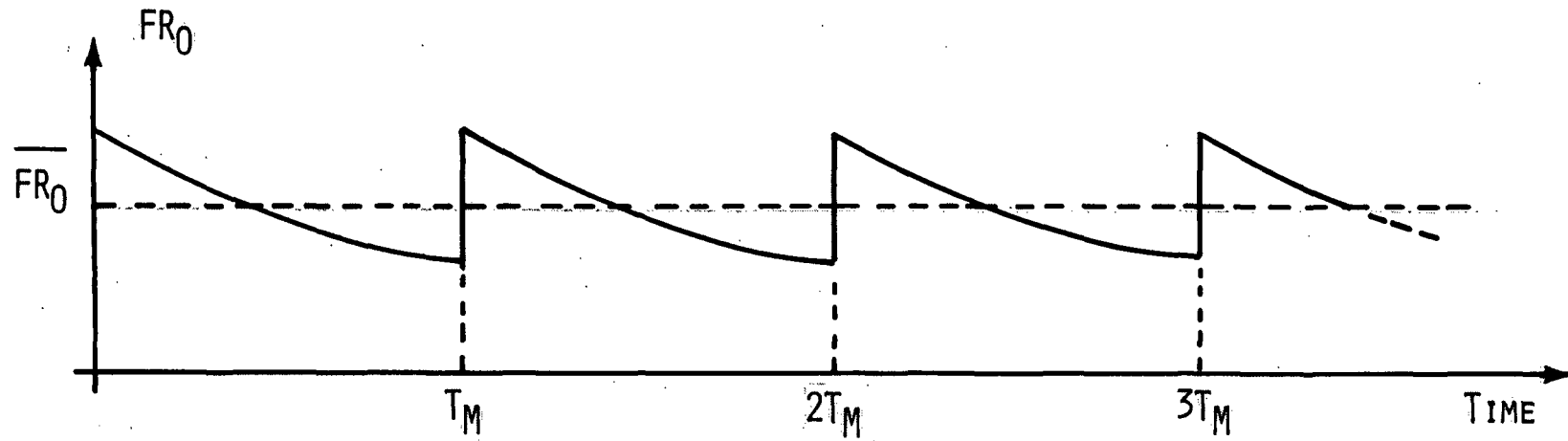
$$FP(t, T) = \frac{\sum_{i=1}^n FR_i(t) \cdot FP_i(T)}{FR(t)}$$

CARSRA evaluates $FP(t, T)$ for two different failure modes and any selected t, T pair, as well as the functional readiness $FR(t)$.

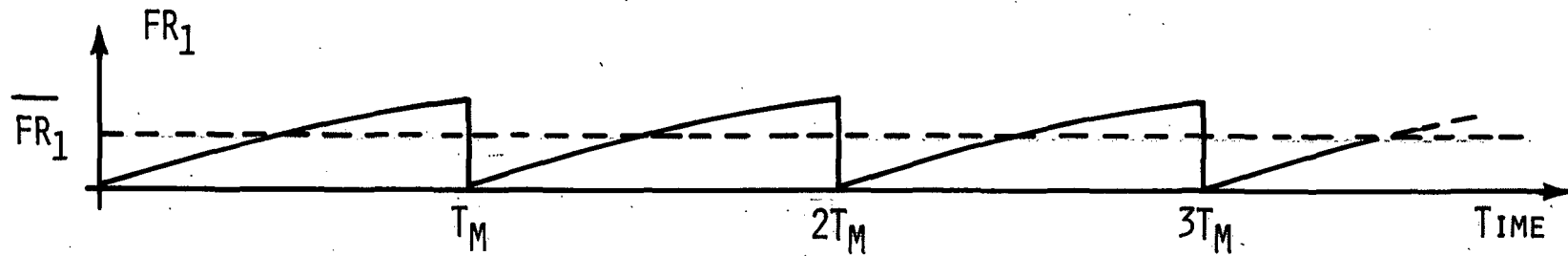
Functional readiness is strictly a system property which assumes that the system is fault free at time zero and that no maintenance action is performed in the interval $[0, t]$.

Functional availability differs from functional readiness in that maintenance is taken into account. The availability has a constant value as soon as a certain maintenance schedule is specified. It may be interpreted as being the *average functional readiness* given a prescribed maintenance strategy. To clarify this, assume for simplicity a maintenance strategy based on periodic maintenance with a maintenance interval T_M . The probability of having the system operational at a certain redundancy level, i.e., the functional readiness, will vary between each maintenance time. Figure 57 displays the probabilities of (a) no module failure (FR_0) and (b) a single module failure (FR_1) as a function of time. The availability, AV , of the system assuming a functional readiness criterion, equivalent to all modules operational *or* a single module failure, is then the average:

$$AV = \frac{1}{T_M} \int_0^{T_M} (FR_0(t) + FR_1(t)) dt = \overline{FR_0} + \overline{FR_1}$$



(a) No Module Failure



(b) Single Module Failure

Figure 57.—Functional Readiness Variations During Scheduled Maintenance

The system failure probability, given the above defined functional readiness criterion and maintenance strategy, is given by:

$$\overline{FP}(t) = \frac{1}{T_M} \int_0^{T_M} FP(t, \tau) dt$$

ARCS Reliability Study Components.—Two major tasks must be carried out before the CARSRA input data may be identified. The first is to establish the system model, and the second is to assess the model parameters. An overview of CARSRA reliability assessment components is given in figure 58.

An important assumption for the ARCS reliability study concerns the effect of power failures. Aircraft power buses, in general, exhibit large failure rates (on the order of 0.5×10^{-3} /hour). This failure rate will usually be larger than the failure rates of most of the equipment served by the bus. In systems where a high level of reliability is demanded, bus redundancy is often employed whereby the function of a bus is backed up by another bus or power source. In effect, power bus redundancy will make each power bus fail-operational so that it takes two consecutive failures before power on any one bus is lost, making the effective failure rate of the bus negligible in comparison to that of the equipment served by the bus. Possible ways of implementing bus redundancy are:

- Switching between buses
- Backup batteries
- OR-ing of dc buses
- Dual ac inputs

All of these approaches have been used in the past, and it is safe to assume that redundancy will be designed into any new commercial transport power system. For the ARCS reliability study, the assumption was therefore made that the influence of power failures may be ignored.

The system modeling task begins with partitioning the system into stages where a stage, in the ARCS list of definitions, has been defined as being a set of identical redundant modules. The second step consists of defining the dependency structure in the form of a dependency tree diagram, which identifies all dependencies of the type “when module A fails, it will cause loss of function of modules B, C, D, etc.” A Markov model is then established for every stage, describing the possible operational states of the stage and stage failure states (detected and undetected).

System dependency tree diagrams for the ARCS near-, intermediate-, and far-term application models are shown in figures 59 through 61. The interpretation of this type of representation was described above. Each stage is represented by a block. The significance of the numbers associated with the blocks is as follows:

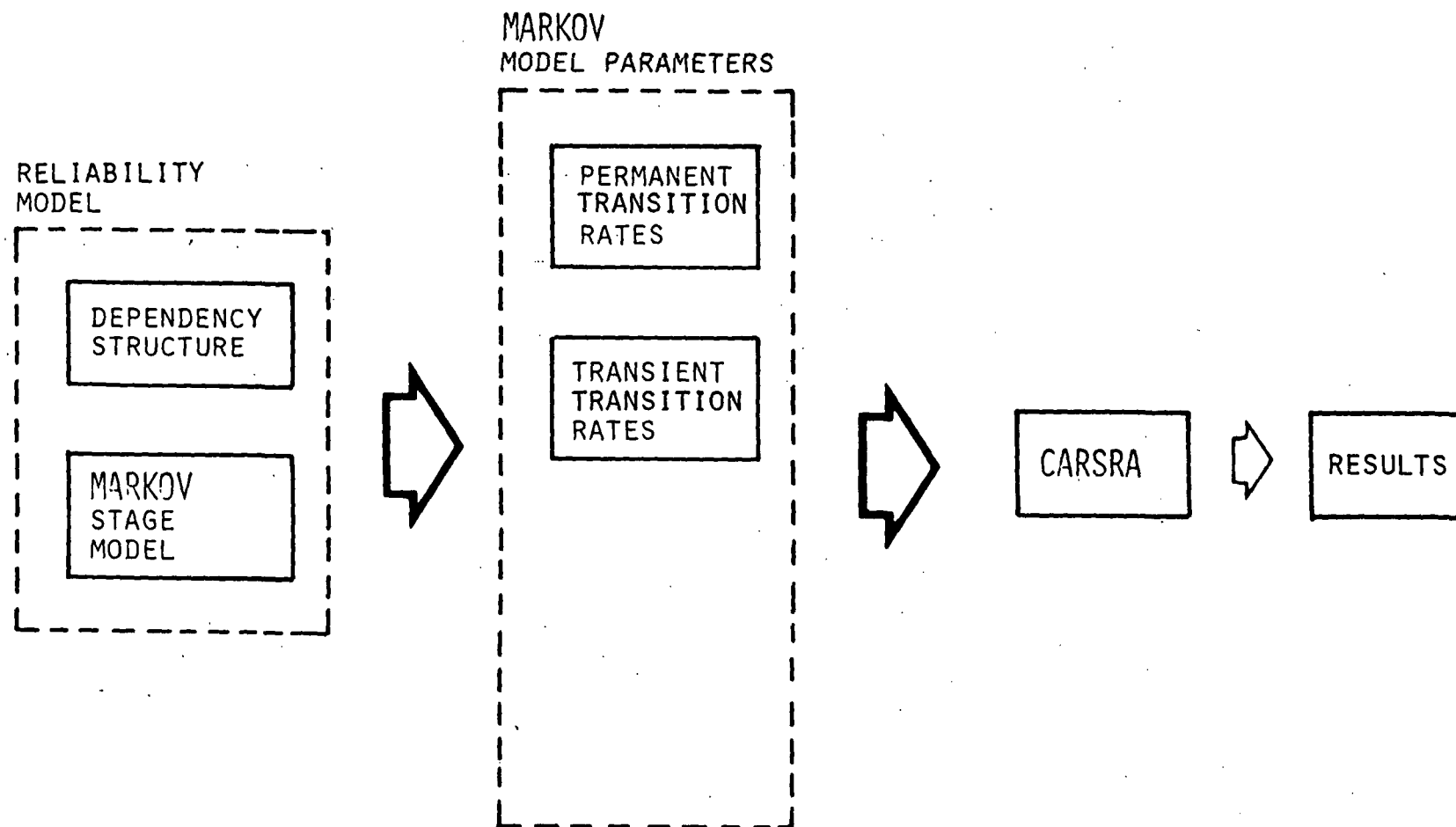
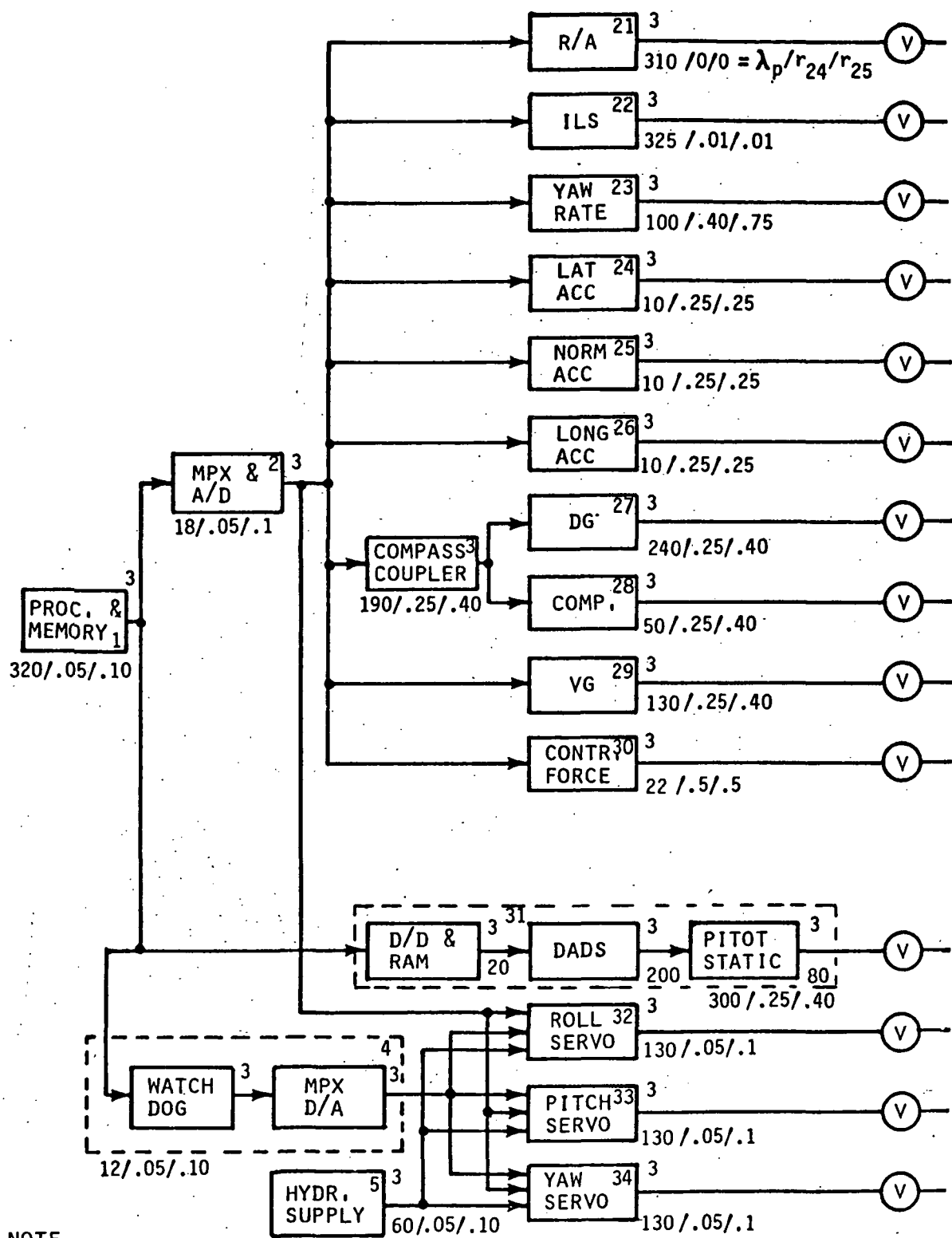


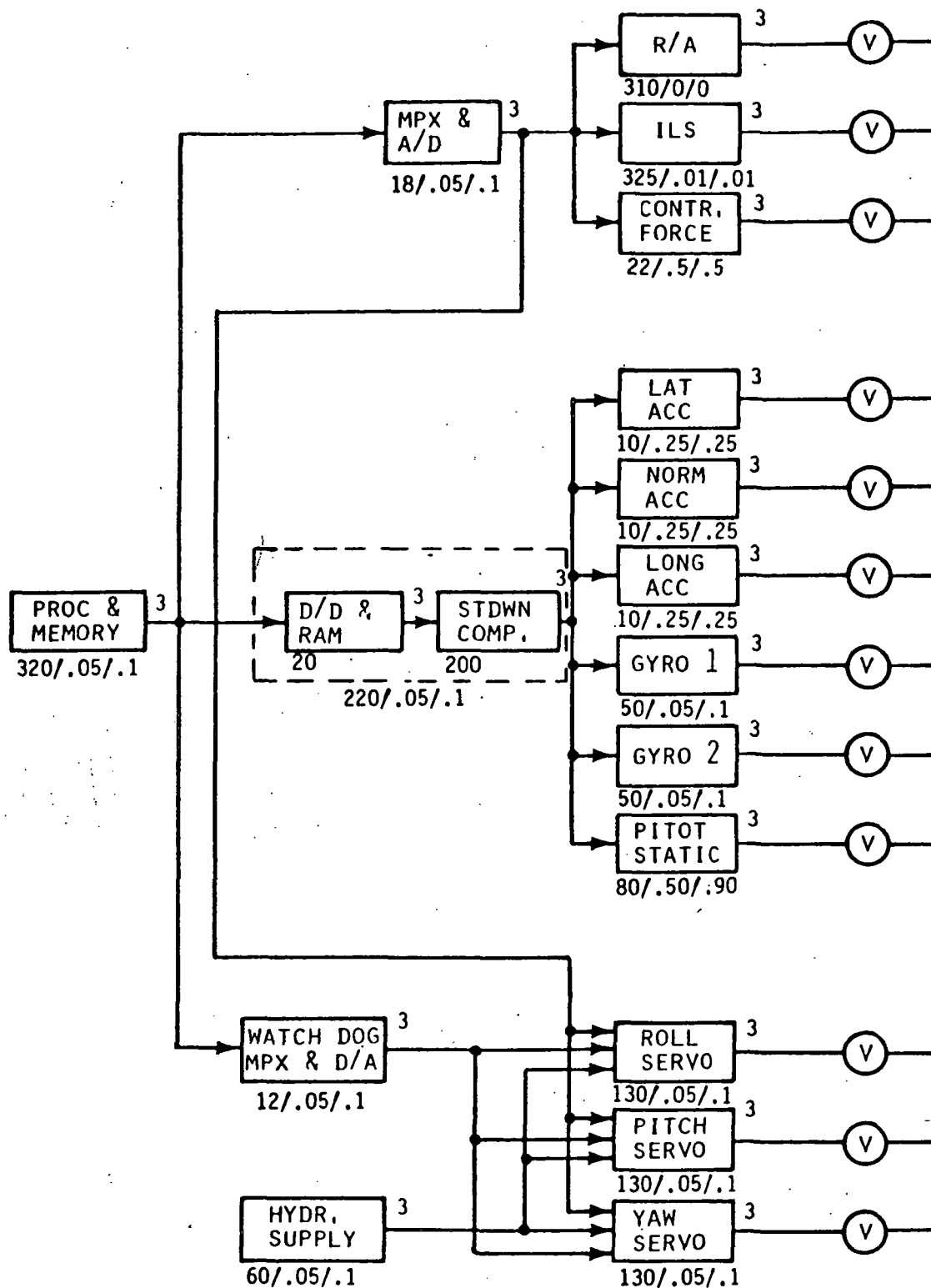
Figure 58.—Reliability Study Components



NOTE

FOR CWS, DELETE R/A AND ILS

Figure 59.—Near-Term ARCS Dependency Tree



NOTE: FOR CWS, DELETE R/A AND ILS

Figure 60.—Intermediate-Term ARCS Dependency Tree

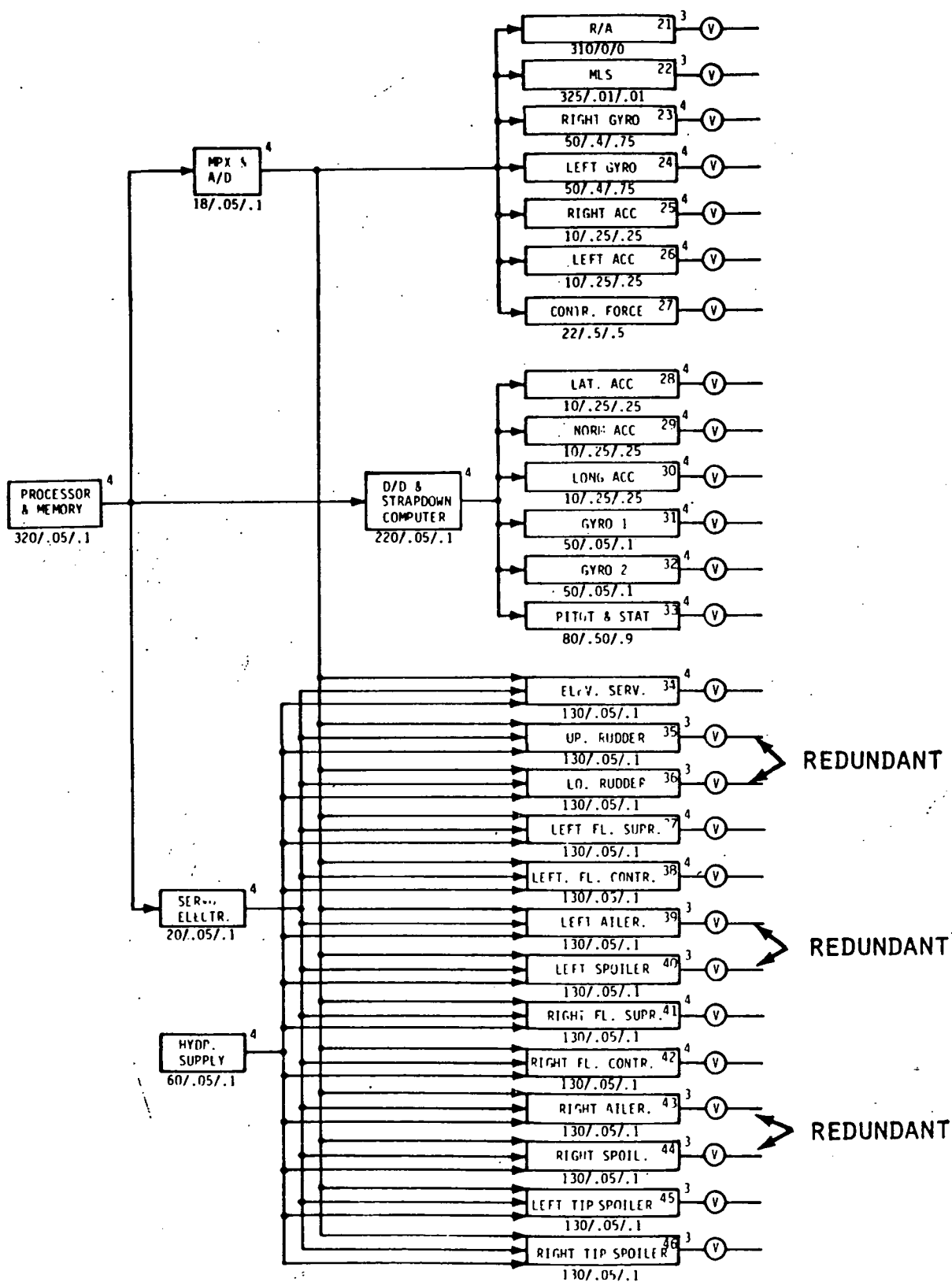
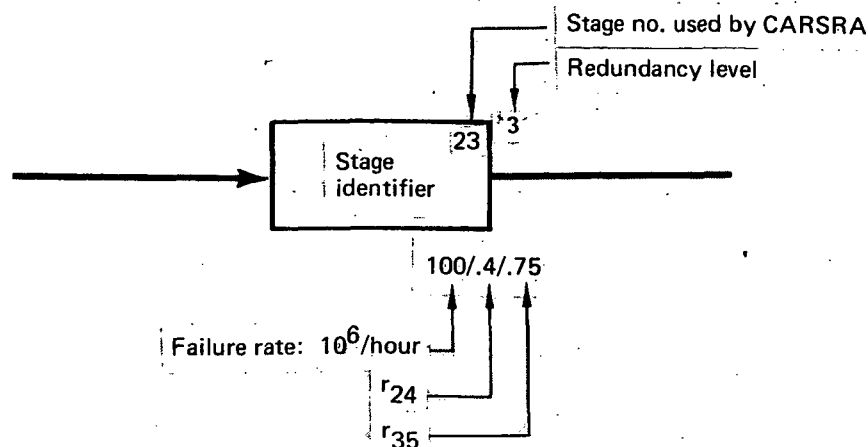


Figure 61.—Far-Term ARCS Dependency Tree



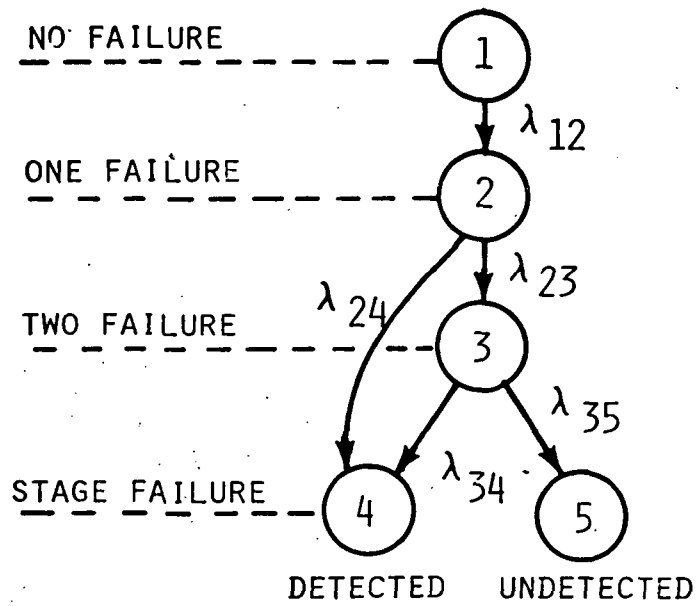
The circles to the right identify system functions all of which may, or may not, be required for a certain operational mode. The assumption is made for the near- and intermediate-term model that all stages but the radio altimeter and ILS are needed for the CWS mode. In the far-term system the following servo functions are assumed to be redundant:

- Upper and lower rudder
- Left aileron and left spoiler (or left tip spoiler)
- Right aileron and right spoiler (or right tip spoiler)

A detailed evaluation of the transition rate ratios r_{24} and r_{35} , a difficult and time-consuming task, was outside the ARCS program scope. Conservative estimates of those parameters based on engineering judgment and several available sources were therefore used in the reliability analysis. These sources and the estimated transition rates are summarized in appendix H.

The Markov model assumed for the near- and intermediate-term ARCS is presented in figure 62. There are five states, three success and two failure states. A crucial assumption implicit in this model is the absence of a single-point failure mode, i.e., the absence of a transition from state 1 to a failed state. Thus the system is assumed to survive all possible single failures, an assumption that has to be carefully verified by a failure mode effect analysis (FMEA). The stage may or may not survive a second failure, as modeled by the transitions from state 2 to 3 and from state 2 to 4, respectively. All second failures are, however, assumed detectable by comparison of similar signals. The rate of transitions between states is specified by λ_{ij} , which is partitioned into a permanent failure component and a transient failure component. The permanent component, $\lambda_{pi} \cdot r_{ij}$, is the product between the exit rate from state i due to permanent failures, λ_{pi} , multiplied by the ratio (or fraction) of these transitions that goes to state j . Similarly, the transient component, $\lambda_{Ti} \cdot l_{ij}$, is partitioned into a transient transition rate, λ_{Ti} , and a "leakage" factor, l_{ij} .

The failure coverage, c_i , for state i , is defined as a sum of rate ratios, r_{ij} , for the transitions to success states j :



$$\lambda_{ij} = \lambda_{pi} \cdot r_{ij} + \lambda_{Ti} \cdot e_{ij}$$

r_{23} = SECOND FAILURE COVERAGE

r_{35} = UNDETECTED FAILURE RATIO

Figure 62.—Near-Term/Intermediate-Term Markov Stage Model

$$c_i = \sum_{j=\text{success states}} r_{ij}$$

In the model of figure 64, we have $c_1 = 1$, $c_2 = r_{23}$, and $c_3 = 0$.

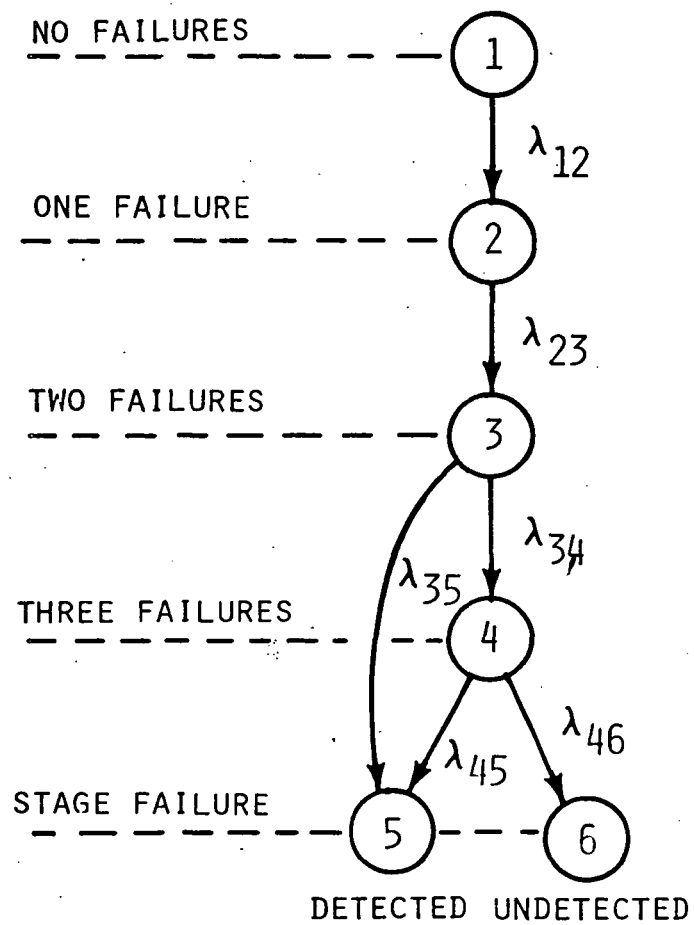
The Markov model for the far-term quadruple stages is shown in figure 63. This model is similar to the triplex-stage model, the only difference being that the quad stage is assumed to survive all possible combinations of two failures.

After establishing the system dependency tree and the Markov models for each stage, the next step is to assess the Markov model parameters, λ_{ij} , which, as was seen above, may be partitioned into permanent failure rates, λ_{pi} ; rate ratios, r_{ij} ; transient fault rates, λ_{Ti} ; and leakages, l_{ij} . For the ARCS analysis, the most significant of these are the coverage parameters r_{23} of figure 62 and r_{34} of figure 63, which model the performance of the ARCS duplex-to-simplex redundancy degradation process. The estimation of the coverage parameters is of fundamental importance in the analysis of an ARCS-type system. Unfortunately this assessment is a nontrivial task, in particular for a digital system, due to the vast number of possible failure modes. However, a promising new approach was developed by GE, in close cooperation with Boeing, during the ARCS program. The essence of this method is described below in section 6.1.2.2. A more comprehensive account is given in appendix G.

A significant effort during the ARCS reliability analysis also went into estimating the transient-related Markov model parameters. Four different sources of transient faults were considered: electrical power system transients, lightning, hydraulic pressure transients, and sensor nuisance failures. Since the latter failure source has been particularly troublesome in contemporary systems, the emphasis in the analysis was directed toward the prediction of sensor nuisance failure rates. It was shown that only a few particular sensors would be troublesome and that the nuisance failure rate could be controlled even for these sensors by proper design of the signal selection and failure-detection algorithms. Since the presentation of this analysis is rather lengthy, it has been deferred to appendix H, which also contains the rate ratio assessments performed for the various stages.

6.1.2.2 A New Coverage Assessment Method

Compared to a triple-modular redundant (TMR) system like the WWCS, the ARCS achieves improved fault tolerance by being able to survive the vast majority of all combinations of two failures because it is able to reconfigure from duplex to simplex operations. The degree of success for this reconfiguration is given by the "coverage for a failure in duplex," which is the conditional probability of system survival given a like failure when the system is operating at duplex redundancy. It was seen above that when the Markov model representation is used, any coverage parameter may be expressed as a sum of rate ratios. In the Markov model representations of figures 62 and 63 for the triplex and quadruplex stage, this sum of rate ratios degenerates to a single rate ratio (r_{23} and r_{34} , respectively). These rate ratios, one for each of the various stages, influence the system failure probability



$$\lambda_{ij} = \lambda_{pi} \cdot r_{ij} + \lambda_{Ti} \cdot e_{ij}$$

Figure 63.—Far-Term Markov Stage Model

very significantly. As a matter of fact, it can be shown that the probability of losing a stage function is proportional to a factor $(1 - r_{23})$ for a triplex stage and $(1 - r_{34})$ for a quadruple stage. From this it is clear that the assessment of coverage parameters is of central importance in the reliability analysis.

The ARCS duplex-to-simplex reconfiguration process has three phases: failure detection, failure localization, and redundancy degradation. The primary means of failure detection is comparisons between like signals. Failure localization may be accomplished by several methods, depending on the nature of each particular stage function. Generally, a combination of hardware and software monitors is used. Redundancy degradation to simplex is achieved by successfully isolating the failed module without significantly disturbing the system function.

Of the three phases described above, failure localization is the most crucial, since the ability to localize a failure generally determines the level of coverage attained. The problem of estimating coverage for a computer is therefore closely related to the capability of localizing failures via software self-test and/or hardware monitoring. There is, however, a significant difference. The capability of localizing a failure is generally given by indicating the percent of all failures that will be detected. Coverage, on the other hand, may be expressed as a ratio between transition rates. The relation between coverage, c , and failure localization percentage, d , is expressed by a simple formula

$$c = d \cdot \frac{\bar{\lambda}_d}{\bar{\lambda}_{tot}} \cdot 10^{-2}$$

where $\bar{\lambda}_d$ is the average failure rate of the localized failures and $\bar{\lambda}_{tot}$ is the average failure rate of the total failure populations.

The expression shows that in order to assess coverage, it is necessary not only to determine whether a certain failure mode is detected by self-test but also to estimate the failure rate of each failure mode.

It was shown (see app. G) that sufficient data is presently not available on failure modes of digital devices. An exact coverage assessment is therefore impossible. However, if the assumption is made that $\bar{\lambda}_d \geq \bar{\lambda}_{tot}$, i.e., that the average failure rates of the covered failures is greater or equal to the failure rate of all failures, coverage could be estimated by

$$c \geq d \cdot 10^{-2}$$

In appendix G it was shown that the assumption $\bar{\lambda}_d \geq \bar{\lambda}_{tot}$ is reasonable if the set of covered failures constitutes a random sample from the set of all possible failures.

With this assumption, the problem of estimating coverage reduces to assessing the portion of all failure modes that will be detected by self-testing. In a digital system this is a formidable task because of the high number of possible failure modes. Even if the analysis is limited

only to “stuck at” failures, the number of failure modes to be considered may often be on the order of tens of thousands. An exhaustive analysis will therefore be impractical, if not impossible.

To circumvent this problem, it was suggested that coverage be estimated from a sample of randomly selected failure modes. The quality of the resulting coverage estimate will then depend on the sample size. A confidence interval could be constructed for the estimate with a width depending on the sample size much like in the theory of random sampling. The idea of assigning a confidence interval to the coverage estimate is in line with the general practice of giving component failure rate data with 60% or 90% confidence.

The approach outlined above was tested by estimating coverage for the ARCS servo state. This stage was selected rather than the computer stage because the servo stage could be analyzed by a conventional manual FMEA. The results of the sampling approach could therefore be compared with FMEA results.

The close agreement found between the two methods supports the validity of basing coverage assessment on a set of randomly selected failure modes using a methodology borrowed from random sampling theory. The reader is referred to appendix G for further details.

6.1.2.3 Study Results

This section contains the major reliability results generated by CARSRA for several system configurations ranging from the WWCS to the far-term ARCS application. Two main elements were involved in the application of CARSRA. The first, and most substantial, was the establishment of the various parameters needed to evaluate each configuration, and the second was the actual running of the program followed by the collection and presentation of results. These results and their relation to stated ARCS requirements and goals will be presented in this section; the rather lengthy and cumbersome considerations that went into the definition of various model parameters are given in appendix H.

Three baseline ARCS application models were established early in the ARCS program effort: the near-term, the intermediate-term, and the far-term applications. These models were to be compared with a control system configured around the WWCS MCP-703 computer system. To provide an equal basis for comparison, the WWCS was assumed to use sensor and servos identical to those of the near-term ARCS application.

Reliability data generated during the reliability/trade study included the following:

Baseline Studies:

- WWCS and ARCS baseline near- and intermediate-term CWS mode system failure probabilities
- WWCS and ARCS near- and intermediate-term autoland mode availability and failure probabilities
- Probability of a diversion and system failure probability for the ARCS far-term application

Trade Studies:

- Reliability data for four different voting node configurations
- Transition rate ratio sensitivities for three ARCS near-term voting configurations
- Latent sensor failure sensitivity for the ARCS near-term application
- ARCS cross-channel communication link study

The results and conclusions from these studies, except the cross-channel communication study, are presented below. (Appendix I contains the cross-channel communication study.) It is felt that the numbers predicted are conservative since the coverage parameters assumed for the various system modules (app. H) reflect what presently is generally obtainable without resorting to sophisticated monitoring schemes.

CWS Mode Results.—The data presented below are based on the reliability models and parameters of appendix H. Transient failure rates, mainly sensor nuisance failures, were assumed to be the same for the ARCS near-term application and WWCS since these systems have identical sensors. In table 6, CWS mode failure probabilities are listed for the WWCS

Table 6.—Near- and Intermediate-Term CWS Failure Probabilities

Stated ARCS goal: A system failure probability of less than 10^{-5} * for a 1-hour flight

Mission time, hours	WWCS	ARCS near term	ARCS intermediate term
1	2.7×10^{-6}	5.4×10^{-7}	1.7×10^{-7}
2	1.1×10^{-5}	2.2×10^{-6}	6.9×10^{-7}
3	2.4×10^{-5}	4.9×10^{-6}	1.5×10^{-6}
4	4.3×10^{-5}	8.6×10^{-6}	2.8×10^{-6}
5	6.7×10^{-5}	1.4×10^{-5}	4.3×10^{-6}
6	9.6×10^{-5}	2.0×10^{-5}	6.2×10^{-6}
7	1.3×10^{-4}	2.7×10^{-5}	8.4×10^{-6}
8	1.7×10^{-4}	3.5×10^{-5}	1.1×10^{-5}
9	2.2×10^{-4}	4.4×10^{-5}	1.4×10^{-5}
10	2.7×10^{-4}	5.4×10^{-5}	1.7×10^{-5}

* 10^{-5} = "remote" event (FAA)

system and the ARCS near- and intermediate-term systems. A failure probability of 10^{-5} for a 1-hour mission was established as a goal in the ARCS requirement specification. This number is associated with the FAA definition of an improbable event. All failure probabilities were computed assuming a failure-free system at time zero.

From table 6 the conclusion is drawn that all three systems satisfy the stated ARCS goal of a failure probability of less than 10^{-5} for a 1-hour flight.

The ARCS near-term system exhibits a five-fold failure probability improvement relative to the WWCS. This improvement results mainly from the ARCS ability to survive most two like failures. Advanced sensors in the ARCS intermediate-term system reduce the failure probability by another factor of 3. The assumption was made that all stages except the R/A and ILS were required for CWS operation.

Autoland Mode Results.—Autoland failure probabilities were generated for the WWCS, the ARCS near-term, and the ARCS intermediate-term applications using the system models presented above. For the purpose of reliability analysis, a landing was separated into two phases: a 15-second landing phase followed by a 30-second rollout phase. These phases were assumed to require different hardware. The 15-second landing phase was assumed to use all modules except the DG, compass, and compass coupler in the near-term ARCS and WWCS applications, but all modules were required for ARCS intermediate-term application. Furthermore, it was assumed that the rollout phase requires only those modules necessary for the directional control of the aircraft. For the WWCS and the ARCS near-term application, the compass coupler, R/A, normal and longitudinal accelerometers, DG, compass, and air data computers were eliminated from consideration during the rollout phase. In the intermediate application, the R/A, normal and longitudinal accelerometers, one gyro, and pitot/static source were eliminated during the rollout phase.

The resulting autoland failure probabilities are as follows:

<u>WWCS</u>	<u>ARCS near term</u>	<u>ARCS intermediate term</u>
18×10^{-11}	1.7×10^{-11}	0.3×10^{-11}

The numbers are based on the assumption that all modules are functional at Cat III alert height (50 feet). According to the regulations, a hazardous event during landing must be "extremely improbable," which has been interpreted to mean having a probability of occurrence less than 10^{-9} /landing. It may be shown that this figure is consistent with a failure probability requirement of 10^{-7} /landing (sec. 4.1). The conclusion may then be drawn that all three systems considered satisfy the requirement by a wide margin, provided all modules are functional at minimum decision height.

The requirement that all system modules are to be functional at minimum decision height imposes a severe constraint on the availability of the autoland function since the MTBF of a redundant system of this complexity is quite low. The MTBF is 125 hours for the WWCS and 137 hours for the ARCS near-term application.

Requiring that all modules be functional places them, from a reliability point of view, in series, so that the probability of having no failures may be estimated by:

$$p(\text{no failure at } t) = e^{-\lambda t}$$

with $\lambda = 1/125$ for the WWCS and $\lambda = 1/137$ for the ARCS near-term application.

Table 7 displays the probability of no module failure as a function of time. The very rapidly decreasing functional readiness of the Cat III autoland function is a definite disadvantage of the WWCS system *as well as of any complex redundant system that requires all modules to be operational at the beginning of a certain critical phase of the mission.*

Table 7.—Probability of No Module Failure

Time, hours	WWCS	ARCS near term
20	0.85	0.86
40	0.73	0.75
60	0.62	0.64
80	0.52	0.56
100	0.45	0.48

The ability of the ARCS to survive the vast majority of all combinations of two failures suggests the possibility of permitting a Cat III landing even if a module is failed in the system. The implication of this possibility will be demonstrated next.

It was pointed out earlier that the functional readiness (and functional availability) of a redundant Cat III autoland system may be improved by permitting landing with degraded redundancy. However, this has to be accomplished without violating the system failure probability requirement of 10^{-7} /landing. ARCS near- and intermediate-term systems were analyzed assuming the following six selected functional readiness criteria:

1. No module failed
2. No module or any one sensor failed
3. No module or any one computer failed
4. No module or any one servo failed
5. No module or any one sensor or servo failed
6. No module or any one sensor, computer, or servo failed

Of the six criteria, only the first will be acceptable for the WWCS, since this system by definition always fails upon a second like failure.

Autoland availability and failure probability are displayed in tables 8 and 9 for the near- and intermediate-term ARCS applications for scheduled maintenance intervals ranging from $T_M = 20$ hours to $T_M = 100$ hours.

Since CARSRA computes functional readiness which is a system property, rather than availability which depends on the maintenance strategy, the availability numbers of tables 8 and 9 had to be evaluated by hand calculations using the formulas derived below.

Letting R_i be the reliability of a module in stage i , the system availability, assuming a scheduled maintenance interval T , is with all stages surviving:

$$AV = AV_0 = \frac{1}{T} \int_0^T (\prod_i R_i^3) dt = \frac{1}{T} \int_0^T e^{-3 \sum_i \lambda_i t} dt$$

which with $3 \sum_i \lambda_i = \lambda_s$ becomes:

$$AV_0 = \frac{1}{\lambda_s T} (1 - e^{-\lambda_s T})$$

The availability corresponding to having all stages operational or one module failed in stage j is $AV = AV_0 + AV_j$ where AV_j is given by:

$$\begin{aligned} AV_j &= \frac{1}{T} \int_0^T (\prod_{i \neq j} R_i^3) 3 R_j^2 (1 - R_j) dt = \frac{1}{T} \int_0^T (\prod_i R_i^3) \cdot 3 \cdot R_j^{-1} \cdot (1 - R_j) dt \\ &= \frac{3}{T} \int_0^T (e^{-(\lambda_s - \lambda_i)t} \cdot e^{-\lambda_s t}) dt = \frac{3}{T} \cdot \frac{1}{(\lambda_s - \lambda_i)} \cdot [1 - e^{-(\lambda_s - \lambda_i)T}] - 3AV_0 \end{aligned}$$

This formula is easily generalized to availability criteria corresponding to one module failed in several stages:

$$AV = AV_0 + \sum_i AV_i$$

$$AV = AV_0 + \sum_i \left\{ \frac{3}{T} \cdot \frac{1}{(\lambda_s - \lambda_i)} \cdot [1 - e^{-(\lambda_s - \lambda_i)T}] - 3AV_0 \right\}$$

where the summations are over stages with a failed module.

The results show that the availability of the Cat III autoland function may be improved very significantly by permitting landings with a degraded system.

Table 8.—Autoland Availability/Failure Probability Results for ARCS Near-Term Baseline

MAINTENANCE INTERVAL TM (HOURS)	NO MODULES FAILED		NO MODULES FAILED OR ANY ONE SENSOR FAILED		NO MODULES FAILED OR ANY ONE COMPUTER FAILED		NO MODULES FAILED OR ANY ONE SERVO FAILED		NO MODULES OR ANY ONE SENSOR OR ANY ONE SERVO FAILED		NO MODULES FAILED OR ANY ONE SENSOR OR ANY ONE COMPUTER OR ANY ONE SERVO	
	AVAILABILITY	AVERAGE FAILURE PROBABILITY	AVAILABILITY	AVERAGE FAILURE PROBABILITY	AVAILABILITY	AVERAGE FAILURE PROBABILITY	AVAILABILITY	AVERAGE FAILURE PROBABILITY	AVAILABILITY	AVERAGE FAILURE PROBABILITY	AVAILABILITY	AVERAGE FAILURE PROBABILITY
20	.94	1.7×10^{-11}	.96	$.17 \times 10^{-7}$.95	$.37 \times 10^{-7}$.95	$.02 \times 10^{-7}$.97	$.19 \times 10^{-7}$.98	$.50 \times 10^{-7}$
40	.88	1.7×10^{-11}	.92	$.32 \times 10^{-7}$.90	$.75 \times 10^{-7}$.90	$.03 \times 10^{-7}$.94	$.34 \times 10^{-7}$.97	$.95 \times 10^{-7}$
60	.84	1.7×10^{-11}	.90	$.46 \times 10^{-7}$.87	1.0×10^{-7}	.87	$.05 \times 10^{-7}$.93	$.49 \times 10^{-7}$.95	1.4×10^{-7}
80	.79	1.7×10^{-11}	.87	$.59 \times 10^{-7}$.82	1.5×10^{-7}	.82	$.07 \times 10^{-7}$.90	$.63 \times 10^{-7}$.94	1.6×10^{-7}
100	.75	1.7×10^{-11}	.85	$.70 \times 10^{-7}$.79	1.8×10^{-7}	.79	$.09 \times 10^{-7}$.89	$.75 \times 10^{-7}$.93	1.8×10^{-7}

- AN AUTOLAND SYSTEM FAILURE PROBABILITY OF 10^{-7} PER LANDING IS CONSISTENT WITH AN ACCIDENT RISK OF 10^{-9} PER LANDING

- A DESIRABLE GOAL : 90% AVAILABILITY AT 100 HOURS MAINTENANCE INTERVAL.
(SYSTEM MTBF = 137 HOURS)



= SYSTEM FAILURE PROBABILITY
REQUIREMENT (10^{-7}) NOT SATISFIED.

Table 9.—Autoland Availability/Failure Probability Results for Intermediate-Term Baseline

MAINTENANCE INTERVAL TM (HOURS)	NO MODULES FAILED		NO MODULES FAILED OR ANY ONE SENSOR FAILED		NO MODULES FAILED OR ANY ONE COMPUTER FAILED		NO MODULES FAILED OR ANY ONE SERVO FAILED		NO MODULES OR ANY ONE SENSOR OR ANY ONE SERVO FAILED		NO MODULES FAILED OR ANY ONE SENSOR OR ANY ONE COMPUTER OR ANY ONE SERVO	
	AVAILABILITY	AVERAGE FAILURE PROBABILITY	AVAILABILITY	AVERAGE FAILURE PROBABILITY	AVAILABILITY	AVERAGE FAILURE PROBABILITY	AVAILABILITY	AVERAGE FAILURE PROBABILITY	AVAILABILITY	AVERAGE FAILURE PROBABILITY	AVAILABILITY	AVERAGE FAILURE PROBABILITY
20	.94	$.32 \times 10^{-11}$.96	$.03 \times 10^{-7}$.97	$.20 \times 10^{-7}$.97	$.02 \times 10^{-7}$.99	$.05 \times 10^{-7}$.99	$.23 \times 10^{-7}$
40	.90	$.32 \times 10^{-11}$.94	$.05 \times 10^{-7}$.91	$.40 \times 10^{-7}$.91	$.04 \times 10^{-7}$.95	$.09 \times 10^{-7}$.98	$.43 \times 10^{-7}$
60	.85	$.32 \times 10^{-11}$.91	$.07 \times 10^{-7}$.87	$.55 \times 10^{-7}$.87	$.06 \times 10^{-7}$.93	$.12 \times 10^{-7}$.97	$.60 \times 10^{-7}$
80	.81	$.32 \times 10^{-11}$.89	$.09 \times 10^{-7}$.84	$.75 \times 10^{-7}$.84	$.07 \times 10^{-7}$.92	$.15 \times 10^{-7}$.95	$.73 \times 10^{-7}$
100	.76	$.32 \times 10^{-11}$.86	$.11 \times 10^{-7}$.80	$.93 \times 10^{-7}$.80	$.09 \times 10^{-7}$.90	$.20 \times 10^{-7}$.94	$.84 \times 10^{-7}$

The system failure probability constraint is satisfied in the near-term system for all considered functional readiness criteria provided the maintenance interval does not exceed $T_M = 40$ hours. Longer maintenance intervals may be tolerated by requiring all computers to be operational at alert height. However, this will decrease the availability. For the intermediate-term system, all considered functional readiness criteria will result in acceptably low system failure probability for maintenance intervals not exceeding $T_M = 100$ hours.

The availability improvements realizable by an ARCS-type system, compared to a TMR system like the WWCS, could be a decisive factor in a commercial STOL application where every STOL landing will have safety requirements similar to those of a Cat III automatic landing.

CCV Fly-By-Wire Mode Results.—A controls configured vehicle (CCV) application, where a failure of the control system will result in loss of the aircraft, was assumed for the ARCS far-term system. A requirement that the system failure probability not exceed 10^{-9} for a 1-hour flight and 10^{-6} for a 10-hour flight was postulated in the ARCS requirements.

The baseline far-term system employs quadruple redundancy in all stages except in stages with redundant functions, such as the upper and lower rudder servos and the aileron and spoiler servos, for which triple redundancy was assumed.

In the operational model for the far-term application, it was assumed that a diversion will be made if two like module failures occur during a mission. The duration of a diversion is assumed to be no longer than 30 minutes. The probability of a diversion and the probability of system failure are presented in table 10.

Table 10.—ARCS Baseline Far-Term Results

Scheduled mission time, hours	System failure probability	Diversion probability
1	0.13×10^{-9}	0.15×10^{-5}
2	0.52	0.61
3	0.98	1.13
4	2.07	2.40
5	3.22	3.73
6	4.61	5.34
7	6.23	7.22
8	8.09	9.30
9	10.20	11.80
10	12.50	14.50

The results of table 10 are based on current (1975) failure rates. The conclusion may therefore be drawn that a quadruple control system using ARCS technology will be adequate for a future CCV application. However, it should be pointed out that the reliability model assumes perfect coverage for any combination of two failures (and the vast majority of third failures). The analysis also indicated that the far-term requirements could not be met by a conventional fail-op/fail-op/fail-passive quadruple system.

Voting Node Study Results.—The objective of the voting node study was to compare four different voting configurations with respect to their reliability and feasibility of implementation. The configurations considered are shown in figure 64.

The “brickwall” configuration is characterized by a single voting node located at the secondary actuator output (force voting) but is otherwise identical to the baseline ARCS. Each channel uses the same sensor information exchanged via the cross-channel links but does not vote the sensor signals. Failure of any module (sensor, computer, or servo) in a channel will cause loss of the whole channel function. Since each computer has access to the same data in the brickwall configuration (A) as in the baseline configuration (B), parameters like second-failure coverages and transient leakages will be assumed identical. This also applies to the other two configurations. The result of the voting node study will therefore be to isolate the effect of different voting strategies.

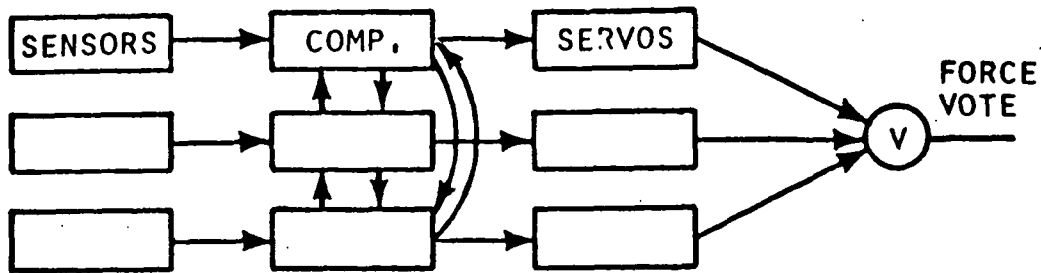
Configuration C is the baseline ARCS with servo command voting added. The servo command signals are crossfed to servo interface modules in each channel, which perform signal selections. The extra hardware required to implement this feature is not insignificant if the ability to degrade from triplex to duplex to simplex servo redundancy is to be preserved. The assumption is made that the near- and intermediate-term systems use an analog voting scheme, while an implementation based on microprocessor technology appears to be a more realistic assumption for the far-term system because of the many servo functions.

Configuration D is C with the addition of cross-strapped sensor signals. In this configuration, each computer will have triplicated sensor interfaces. The number of success paths is increased in configuration D since a computer failure does not imply loss of all sensor functions in the channel.

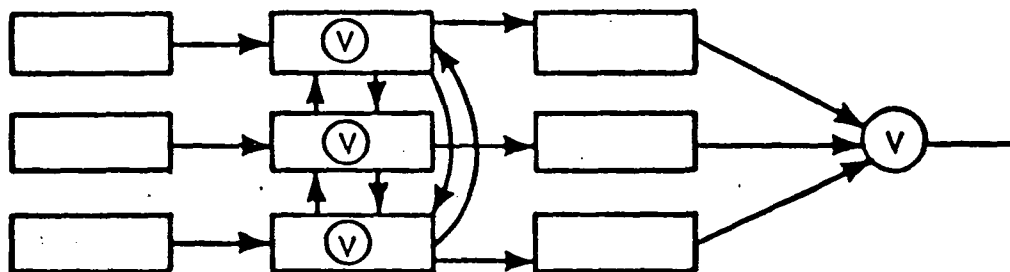
The result of the voting node study is presented in table 11. Several conclusions may be drawn from this table.

1. A large improvement (factor of 5) is obtained by voting the sensor signals.
2. The additional complication of voting the servo command signals is not justified.
3. Cross-strapping the sensor signals is a good idea unless the disadvantage of the additional interface hardware and wiring makes it unattractive. The integrated strap-down air data system (ISADS) used for intermediate- and far-term applications appears to be better suited to sensor cross-strapping than the conventional system used in the near-term application.
4. Voting is more effective in a quadruplex system than in a triplex system.

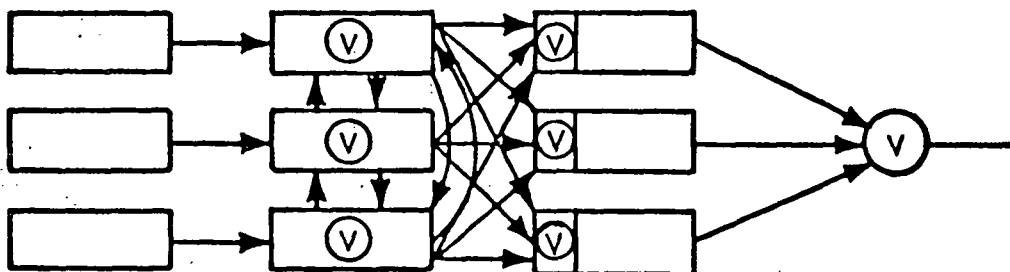
A BRICKWALL (ARCS WITHOUT SENSOR VOTING)



B SENSOR VOTE USING CROSS-CHANNEL LINKS (ARCS BASELINE)



C PLUS SERVO COMMAND VOTING WITH MODE SWITCH



D SENSOR DATA CROSS-STRAPPING AND SERVO COMMAND VOTING WITH MODE SWITCH

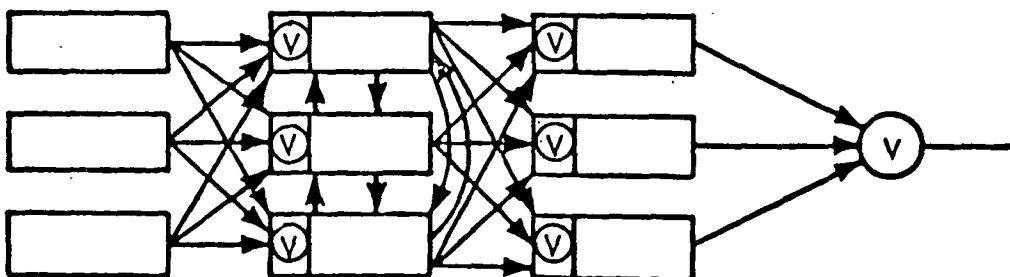


Figure 64.— Voting Trade Study Configurations

Table 11.—ARCS Voting Node Trade Study Results for the CWS Mode Application

Application model	Voting configuration failure probability (1-hour mission)			
	A Brickwall	B Baseline ARCS	C With servo voting	D With sensor and servo voting
Near term	2.6×10^{-6}	5.4×10^{-7}	5.3×10^{-7}	2.3×10^{-7}
Intermediate term	1.3×10^{-6}	1.7×10^{-7}	1.6×10^{-7}	9.6×10^{-8}
Far term	1.9×10^{-9}	1.6×10^{-10}	1.3×10^{-10}	4.8×10^{-11}

Sensitivity Study Results.—The sensitivity study had three objectives: (1) to assess the sensitivity of the near-term system failure probability to variations in Markov model transition rates for the different stages, (2) to assess the relative impact of computer unreliability on system failure probability, and (3) to assess the significance of latent sensor failures.

An example of a Markov model for a stage was shown in figure 62. The parameters λ_{ij} in the figure are the transition rates. These rates and the structure of the Markov model will, together with the dependency between stages, completely specify the system failure probability. The sensitivity of this probability to variations in the rates λ_{ij} could be studied, in principle, by making repeated computer runs with slightly altered values for λ_{ij} . However, this approach would be practical only in a limited study involving a few specific, selected rates.

Another possibility would be to derive an analytical expression for the system failure probability as a function of the rates λ_{ij} . This possibility was explored. An approximate method for evaluating Markov state probabilities was developed. This method was useful not only for generating sensitivity data, but also for checking CARSRA program outputs for gross errors. In its simplest form, the approach finds the term of lowest order in t in the Taylor expansion for the Markov state probabilities. This term provides a good estimate of the actual state probability if $\lambda_{ij}t \ll 1$ for all λ_{ij} . The essence of the approach will now be outlined.

Consider the simple Markov model of a triplex stage shown in figure 62. The Markov equation for state 2 is:

$$\dot{p}_2 = -(\lambda_{23} + \lambda_{24}) p_2 + \lambda_{12} p_1$$

If for $t = 0$, $p_1 = 1$, and $p_2 = 0$, we will have $p_1 \gg p_2$ for small t . Therefore

$$\dot{p}_2 \approx \lambda_{12} p_1$$

which after integration becomes

$$p_2 = \lambda_{12} t + O(t^2)$$

Similarly, for state 3:

$$\dot{p}_3 = -\lambda_{34} p_3 + \lambda_{23} p_2$$

which, if $p_3 \ll p_2$ leads to

$$p_3 = \lambda_{12} \lambda_{23} \frac{t^2}{2} + O(t^3)$$

Finally, for state 4:

$$\dot{p}_4 = \lambda_{24} p_2 + \lambda_{34} p_3 = \lambda_{12} \lambda_{24} t + \lambda_{12} \lambda_{23} \frac{t^2}{2} + O(t^2)$$

so that

$$p_4 = \lambda_{12} \lambda_{24} \frac{t^2}{2} + O(t^3)$$

The following observations are made from these results. State 2 is reached after *one* transition (from state 1 to state 2) and the approximate probability for this state is the product of the *one* transition rate λ_{12} and t to the power of *one*. State 3 was reached after *two* transitions, and the probability was formed by multiplying the *two* transition rates in the path leading from state 1 to state 3 with each other and then multiplying by $t^2/2$. These observations are special cases of the following rule:

The probability of an arbitrary state may be approximated by summation of the contributions from paths leading to the state starting at the source state (state 1 in the example). The contribution from each path is formed by multiplying together the transition rates in the path and then multiplying by $(t^n/n!)$ where n is the number of transitions in the path.

This simple rule makes it possible to find approximate failure probabilities for rather complex systems by hand calculation. As an example, an approximate prediction of the probability of a detected failure in the ARCS near-term system can be made as follows.

The various stages in the ARCS near-term application are listed in table 12 together with a stage number assignment that will be used for this example. The dependency structure of the ARCS near-term system was defined in figure 59. The probability of a detected system failure may be estimated by considering all combinations of two module failures leading to system failure, i.e., all two-step transitions in a comprehensive Markov model that end up in the detected failure state. It is recognized that an error will be made by ignoring combinations of three or more failure combinations equivalent to system failure. This error will be small, however, if $\lambda t \ll 1$ for all transition rates.

All two-step transitions leading to system failure are listed in table 13. The (stage) location of the first transition (which is from the first to the second state in the stage Markov model of fig. 62) is indicated in the first column. Stages for which a second transition (between states 2 and 4 of the stage Markov model) will result in system failure are indicated in the second column. Note that stages 6 and 7 are not required for the CWS mode. The system failure probability may be estimated by summing over all two-transition contributions, $\lambda_{12}^{(i)} \cdot \lambda_{24}^{(j)} \cdot t^2/2$, where i and j pertain to stage numbers in columns 1 and 2 respectively.

$$\begin{aligned}
FP(t) = & \left[\lambda_{12}^{(1)} \left(\sum_{i=1}^4 \lambda_{24}^{(i)} + \sum_{i=7}^{18} \lambda_{24}^{(i)} \right) + \lambda_{12}^{(2)} \left(\lambda_{24}^{(2)} + \lambda_{24}^{(4)} + \sum_{i=7}^{14} \lambda_{24}^{(i)} \right. \right. \\
& + \left. \sum_{i=16}^{18} \lambda_{24}^{(i)} \right) + \lambda_{12}^{(3)} \left(\lambda_{24}^{(3)} + \sum_{i=16}^{18} \lambda_{24}^{(i)} \right) + \lambda_{12}^{(4)} \left(\lambda_{24}^{(4)} \right. \\
& + \left. \sum_{i=11}^{12} \lambda_{24}^{(i)} \right) + \sum_{i=7}^{18} \lambda_{12}^{(i)} \lambda_{24}^{(i)} + \lambda_{12}^{(19)} \left(\lambda_{24}^{(19)} + \sum_{i=16}^{18} \lambda_{24}^{(i)} \right) \left. \right] \frac{t^2}{2}
\end{aligned}$$

Evaluating this expression using the failure rates defined in figure 59 yields

$$FP(t) = 0.547 \cdot 10^{-6} \cdot t^2$$

where t is the exposure time in hours. Good agreement is shown with the second column of table 6.

Table 12.—Definition of Stage Numbers for the ARCS Near-Term Application

Stage	Stage no.
Processor and memory	1
Multiplex and A/D	2
Watchdog and multiplex D/A	3
Compass coupler	4
R/A	5
ILS	6
Yaw rate	7
Lateral accelerometer	8
Normal accelerometer	9
Longitudinal accelerometer	10
DG	11
Compass	12
VG	13
Control force sensor	14
Air data computer	15
Roll servo	16
Pitch servo	17
Yaw servo	18
Hydraulic supply	19

**Table 13.—Two-Step Failure Transitions Leading to
System Failure for the ARCS Near-Term
CWS Mode Application**

First failure transition (λ_{12}) in stage	Second failure transition leading to system failure (λ_{24}) in stages
1	1, 2, 3, 4, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18
2	2, 4, 7, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18
3	3, 16, 17, 18
4	4, 11, 12
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19	16, 17, 18, 19

The method outlined was used to establish and approximate an analytical expression for the probability of a detected failure for the ARCS near-term system using all stages. It was seen above that this expression is a second-order polynomial in t with terms of the type $\lambda_{12}^{(k)} \cdot \lambda_{24}^{(j)} \cdot t^2/2$, where k and j refer to separate stages.

The desired sensitivity factors could then be derived by differentiations. Two types of sensitivity factors were used:

1. Transition rate sensitivity, $S_{ij}^{(k)}$, defined by

$$S_{ij}^{(k)} = \frac{\delta (FP)}{\delta \lambda_{ij}^{(k)}} \cdot \frac{100}{FP}$$

The factor $S_{ij}^{(k)}$ expresses the *percent* change in the failure probability caused by a change of $\Delta \lambda_{ij}^{(k)} = 1/\text{hour}$ where $\lambda_{ij}^{(k)}$ is the transition rate from state i to state j in stage k . Thus we have

$$\Delta FP \% = S_{ij}^{(k)} \Delta \lambda_{ij}^{(k)} \%$$

2. Rate ratio sensitivity, $R_{ij}^{(k)}$, defined by:

$$R_{ij}^{(k)} = \frac{\delta(FP)}{\delta r_{ij}^{(k)}} \cdot \frac{100}{FP}$$

A variation, $\Delta r_{ij}^{(k)}$, will cause a failure probability change of ΔFP %:

$$\Delta FP \% = R_{ij}^{(k)} \cdot \Delta r_{ij}^{(k)} \%$$

The rate ratio sensitivity factor is mainly of interest when assessing the sensitivity to second-failure coverage $c = r_{23} = 1 - r_{24}$ in the various stages.

The results of the sensitivity analysis are summarized in table 14, where the factors S_{12} , S_{24} , and R_{24} are presented for the various stages and the voting configurations B, C, and D of figure 64.

Several interesting observations may be made regarding table 14. Looking at the sensitivities S_{12} , for example, we find that the baseline ARCS is very sensitive to computer transients that cause a redundancy degradation from triplex to duplex. Assuming, for example, that the computer would be unable to recover from 1% of the abnormal electrical power transients occurring with a rate of 10^{-2} /hour, the resulting increase in failure probability becomes:

$$\Delta FP \% = 5.4 \times 10^4 \cdot 3 \times 10^{-2} \cdot 10^{-2} = .16 \%$$

With sensor and servo cross-strapping, on the other hand, the sensitivity to leaky computer transients is much smaller:

$$\Delta FP \% = 0.78 \times 10^4 \cdot 3 \times 10^{-2} \cdot 10^{-2} = 2 \%$$

The reason for this difference is of course that a loss of a computer in the baseline ARCS causes degradation to duplex in all stages.

For the baseline ARCS, second-failure coverage sensitivities (R_{24}) are most significant for the R/A, ILS, DG, and servo stages, whereas a high computer coverage is most important in the configuration with cross-strapped sensor and servo signals.

It is of interest to find that proportion of system failure probability attributable to computer unreliability. This can be done by letting the transition rates for the computer stages be zero in the CARSRA program. The result is presented in table 15.

The dramatic decrease when introducing sensor cross-strapping may be explained by the fact that the bulk of system unreliability is associated with uncovered sensor failures.

The impact of a computer failure will therefore be very significant if it has the effect of reducing all-sensor redundancy from triplex to duplex.

Table 14.—ARCS Sensitivities for the Near-Term Application Model

ARCS stage	Baseline ARCS			Servo command voting			Servo voting and sensor cross-strapping		
	S ₁₂	S ₂₄	R ₂₄	S ₁₂	S ₂₄	R ₂₄	S ₁₂	S ₂₄	R ₂₄
R/A	Small	1.6×10^5	99	Small	1.8×10^5	112	Small	1.6×10^5	100
ILS	5.3×10^2	1.6	104	6.1×10^2	1.9	124	11.4×10^2	1.7	110
Yaw rate	66	1.0	20	75	1.1	22	140	0.52	10
Accelerometers (3 ea)	4.1	0.82	1.6	4.7	0.9	1.8	8.8	0.05	0.10
DG	99	1.9	91	112	2.1	101	208	2.2	105
Compass	21	1.4	14	23	1.6	16	42.8	1.2	12
VG	55	1.1	29	61	1.3	34	114	0.7	18
Control force	18	0.85	3.7	21	0.95	4.2	39	0.11	5.0
DADS/pitot	135	1.0	60	131	1.1	68	251	1.5	84
Servo (3 ea)	11	1.1	29	12	0.63	16	22	1.2	31
Processor/memory	540	0.79	50	502	0.89	57	78	2.4	214
Compass coupler	200	1.3	49	224	1.5	57	418	1.0	38

Table 15.—Contribution of Computer Unreliability to System Failure Probability

ARCS near-term configuration	Contribution, percent
ARCS baseline	57
With servo voting	55
With sensor cross-strapping and servo voting	12

Latent Sensor Failure Study Results.—The objective of this study was to evaluate to what extent latent sensor failures will impact the reliability results, while at the same time demonstrating the capability of CARSRA to handle a more complicated stage model.

Latent sensor failures are mainly passive failures of signals that nominally are close to zero. These types of failures may go undetected for some time, particularly during quiescent flight conditions, before they are detected. Figure 65 illustrates latent sensor failures.

The seven-state Markov model depicted in figure 66 was used to describe latent failure effects. States 1, 3, 5, and 6 model detected failure states while states, 2, 4, and 7 model latent (undetected) failures. State 2 corresponds to one latent failure, state 4 to one latent and one permanent failure, and state 7 to an undetected stage failure. The terms λ_{12} , λ_{27} , λ_{34} , and λ_{57} are latent failure transition rates; λ_{47} consists of both latent and detected failures; the remaining rates except λ_{23} and λ_{45} are associated with detected failures; and λ_{23} and λ_{45} model the latency detection rate, which describes the process of detecting a latent failure by experiencing a deviation between a good signal and the failed one large enough to trigger the threshold detector.

To make a just comparison between the models with and without latent failures, the transition rates between the different failure levels were made identical. This implies that the two models will always yield the same total probability of a system failure; only the proportion between detected and undetected failures will differ.

Latent failure rates for the sensors were assessed based on sensor failure mode data. (The data sources are discussed in app. H.) Latency detection rates were assessed based on flight records of sensor characteristics during quiescent conditions. A latency detection rate of 0.1/hour was assessed for the majority of the sensors.

Results of the sensor latency study indicate that the estimated probability of an undetected failure increases substantially when taking into account the possibility of latent failures. The estimated probability of an undetected failure for the near-term CWS application, for example, was increased from 1.3×10^{-10} to 1.2×10^{-8} for a 1-hour flight. However, this probability is still small compared to the probability of a detected failure (5.4×10^{-7}).

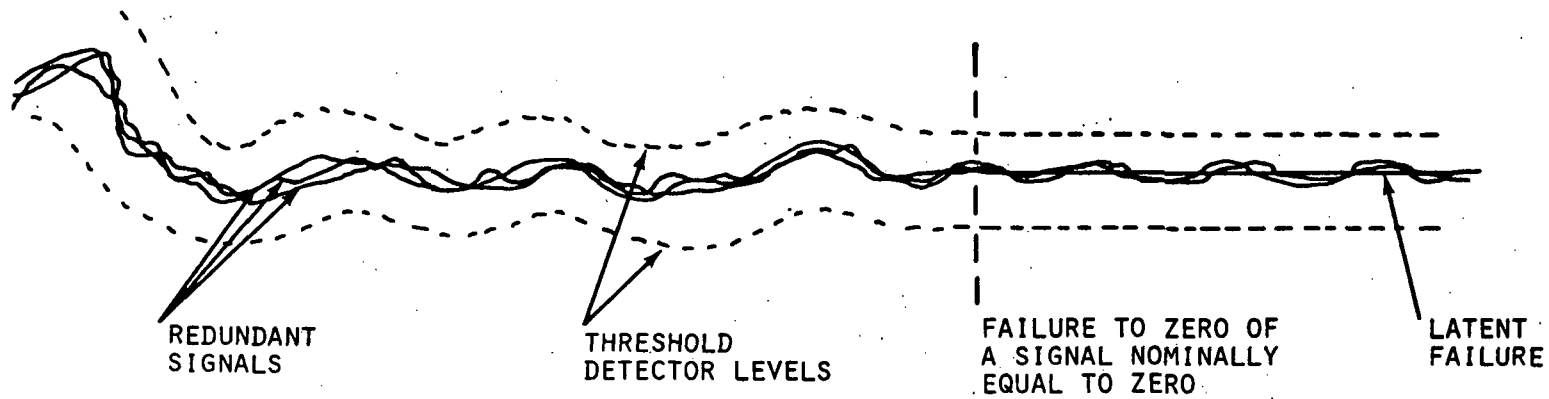


Figure 65.—Latent Sensor Failure Illustration

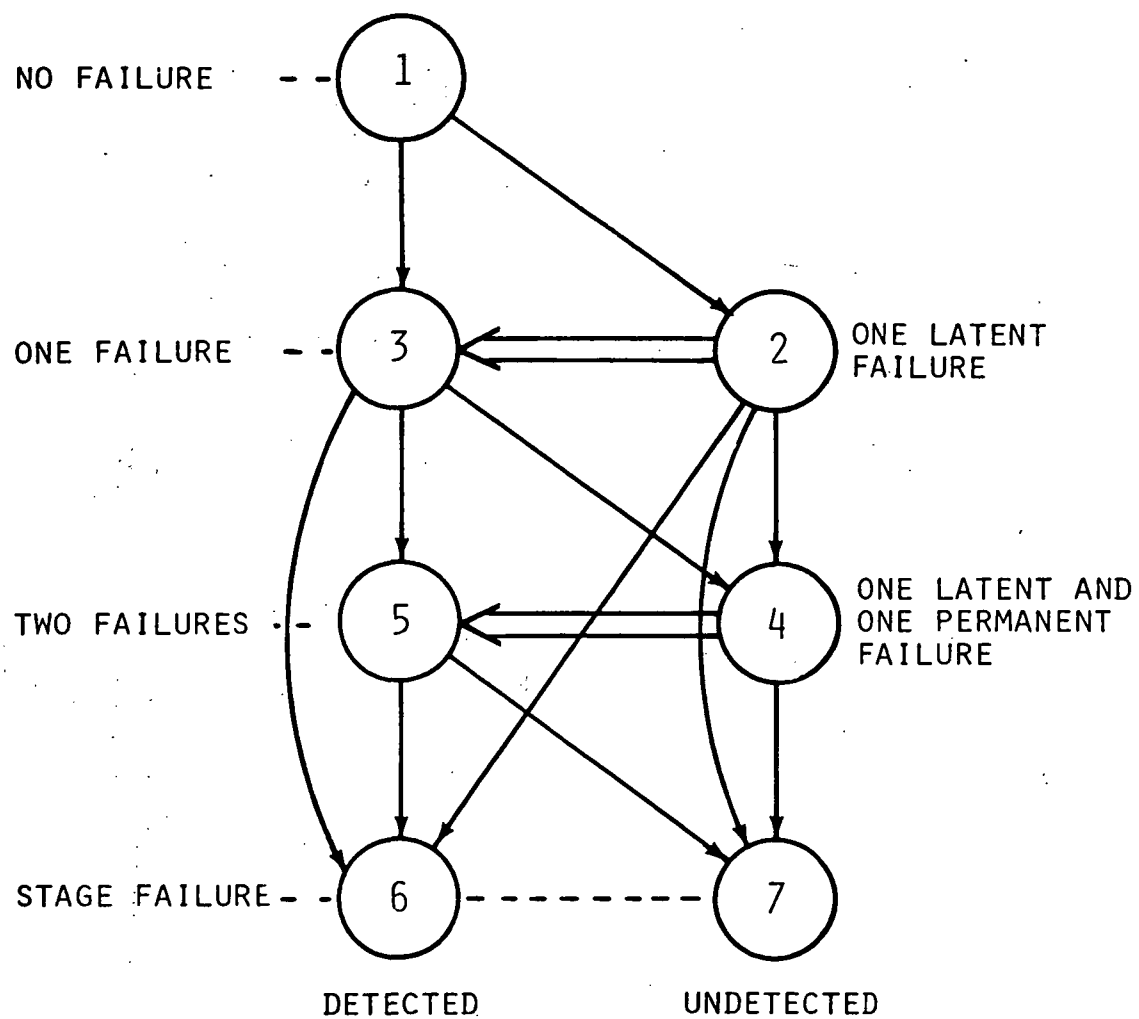


Figure 66.—Latent Sensor Failure Model

6.1.2.4 Reliability Analysis Conclusions

The two main aspects of the ARCS reliability task were to identify a reliability estimation method applicable to the analysis of reconfigurable fault-tolerant systems and to use this tool to estimate the reliabilities of the WWCS and several ARCS candidate systems.

In surveying available reliability estimation tools, it was found that a new approach was needed. Two new reliability estimation programs were developed: RESRA (Redundant System Reliability Analysis) and CARSRA (Computer Aided Redundant System Reliability Analysis). RESRA, which is a forerunner to CARSRA, does not have the same capability as CARSRA but is simpler to use; i.e., it requires less input data and has a shorter processing time.

CARSRA is designed to handle modular-redundant systems that have internal signal consolidation points. It will take into account failure coverage and is able to model the effect of transient faults. It will also model a wide variety of redundancy degradation strategies, including spares augmented systems; and computes functional readiness as well as system failure probability. CARSRA was used extensively to generate the ARCS reliability study results and proved to be a powerful as well as flexible tool particularly suited for configuration trade studies. Its unique capability to compute functional readiness and failure probability greatly facilitated the generation of the data base for the reliability analysis. Appendix F contains a further description of the principle of operation and a user's guide.

The following conclusions were drawn relative to the WWCS and ARCS baseline configurations:

- The WWCS, ARCS near-term, and ARCS intermediate-term systems will all provide adequate control wheel steering mode reliability. However, the ARCS near-term application will exhibit 5 times lower, and the intermediate-term system 15 times lower, failure probability than the WWCS.
- The WWCS, near-term ARCS, and intermediate-term ARCS will also have acceptably low failure probabilities in the Cat III autoland operational mode provided all modules work at alert height.
- The functional readiness of the WWCS Cat III autoland function is 85% after 20 hours operation and 45% after 100 hours operation without maintenance. This poor performance results from the requirement that all modules in the system have to be operational at alert height.
- For the near-term ARCS, the functional readiness may be improved by permitting Cat III landings with one module failed in the system. The resulting functional readiness for the near-term ARCS Cat III autoland function is 97% after 20 hours operation and 89% after 100 hours operation without maintenance. Increased functional readiness is the most important advantage of the ARCS relative to the WWCS.

- The far-term ARCS is a quadruple system designed for CCV operation. Reliability analysis indicates that a quadruple ARCS would satisfy the postulated failure probability requirements.

The ARCS trade study results are summarized as follows.

- Results of the voting node trade study indicate that large reliability improvement is realized by voting the sensor signals but that servo command voting only will result in a small improvement. Also, a system in which each computer directly interfaces with all system sensors via triplicated channel interfaces will exhibit lower failure probability than a system in which sensor data is exchanged via the processor cross-channel links. This added reliability, however, has to be weighed against the additional hardware, which increases the packaging size and complexity and decreases the system MTBF.
- Results of the sensitivity study show that the computer reliability is a dominant factor in configurations where sensor data is transferred via the processor cross channel but is less important if the sensor signals are cross-strapped so that each computer interfaces with all system sensors. The study also shows that the sensors dominate the system reliability, partly because of a high cumulative failure rate (68% of the total system failure rate in the near-term ARCS) and partly because of the conservatively assumed low second-failure coverages.

The main conclusion of the reliability study was that the ARCS capability of redundancy degradation from triplex to duplex to simplex opens up the possibility of initiating a critical flight segment, e.g., a Cat III landing or a STOL takeoff or landing, with a partially degraded system. System availability will thus be dramatically enhanced with the ARCS compared to a TMR system like the WWCS. The observation can then be made that redundancy in the ARCS can be used not only for the purpose of decreasing system failure probability but also as a means of increasing functional availability where degradation to simplex is an acceptable operation.

The observation may also be made for the autoland function that the increased survivability obtained by the ARCS relative to the WWCS does not by itself justify the ARCS since the reliability data presented shows that the WWCS is adequate from a safety point of view. The advantage of ARCS is therefore in the area of improved economy.

6.2 COST/BENEFIT ANALYSIS

An assessment of the overall cost effectiveness of the ARCS was carried out in two phases: an analysis of airline cost-of-ownership for an ARCS maintained in a Category III operational status and an analysis of the cost effect of providing an integrated system test function to augment system maintainability. In general, an avionic system's cost-of-ownership is comprised of three major factors as depicted in figure 67: acquisition cost, maintenance cost, and cost of schedule interruptions caused by the loss of a particular operational capability, e.g., loss of CatII/Cat III approach functions in conjunction with poor weather conditions.

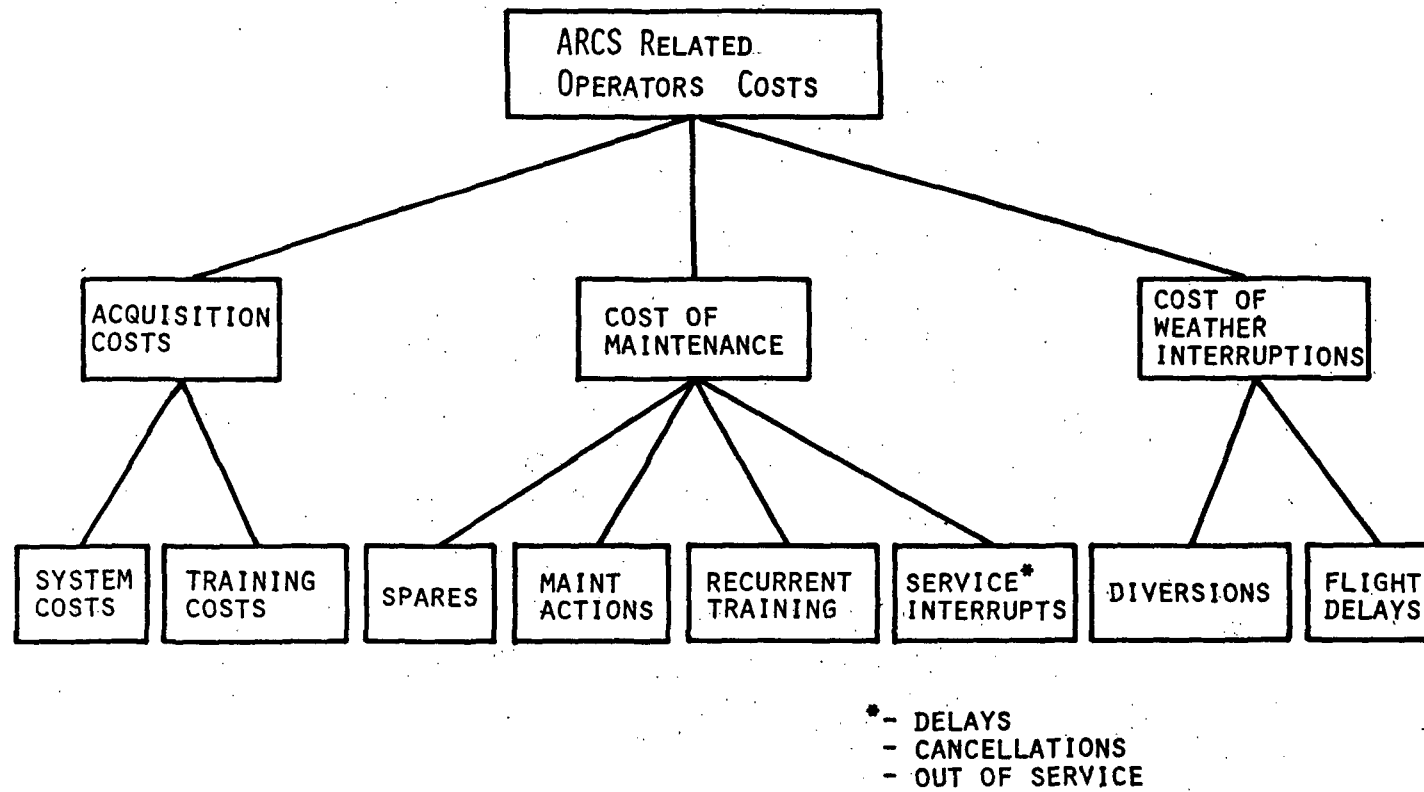


Figure 67.—Avionics Cost Factors

The cost data for the first analysis was compiled by comparing the ARCS to the WWCS, which represents a contemporary triple-modular-redundant (fail-op/fail-passive) system design. The baseline for the second study was the ARCS hardware without the maintenance enhancement elements, maintained in a conventional manner.

The data base was supplied by General Electric and United Airlines. GE provided data relative to the ARCS and WWCS acquisition costs, and United provided cost data relative to an airline's operation as influenced by weather conditions and line and shop maintenance procedures.

An operations model based on the United Airlines 727 operations was selected as a representative framework for assessing cost trends associated with the ARCS technology. The 727 has a route structure that is representative of an average airline having medium trip lengths and is common to many operators. It serves a wide variety of airports throughout the United States including stations with maintenance capability ranging from complete to none. The characteristics of the UA 727 fleet operation of interest for cost-of-ownership computations are summarized below.

Fleet size	150 aircraft
Annual flying time	350 000 hours
Average flight duration	1.3 hours
Airports served	77
Maintenance stations	21

The following discussion is divided into four sections: an analysis of the cost effectiveness of ARCS attributable to its enhanced functional reliability/availability, an evaluation of ARCS acquisition cost, an analysis of the cost effect of providing an integrated system test function, and a summary.

6.2.1 ARCS AVAILABILITY COST EFFECT

The baseline airplane system on which the cost comparisons were made is the ARCS intermediate-term application model. This model reflects a basic command augmentation system, or CWS, and a flight-mandated (required by the operator for dispatch) Cat III autoland capability.

Because the WWCS is a triple-modular-redundant system, any failure within the three channels (be it sensor, computer, or servo) degrades the system below Cat II/III requirements, and it must be repaired prior to a dispatch meeting the operator's revenue operating policy. The ARCS, on the other hand, can be dispatched meeting Cat II/III requirements with a fault, thus deferring system maintenance. The significance of the above baseline considerations were not completely quantified in the study, but are qualitatively brought out, where appropriate, in the cost-related discussions that follow.

6.2.1.1 Maintenance Cost

In evaluating the maintenance cost of the ARCS relative to the WWCS, no consideration was given to the effects of system test. System test effects were dealt with separately and are

assumed to be realizable with either the ARCS or WWCS. Therefore, the primary attribute of the ARCS that makes it more attractive from the standpoint of system maintenance is its ability to continue to operate in a simplex mode. With this capability, the occurrence of any single failure in any stage will not significantly hamper the fault-tolerant characteristics of the system, and hence immediate repair is not required.

This characteristic has several maintenance implications. The number of line stations with system maintenance capability can be reduced since the airplane can continue operations with an existing failure until reaching such a facility, e.g., overnight station. Fewer maintenance stations also implies correspondingly fewer line spares required on a fleet-wide basis. And, since repair of first-failure conditions need not necessarily be made before the next revenue departure, the probability of incurring a maintenance-caused departure delay or cancellation is greatly reduced.

Although viewed as having a potentially significant impact on system cost-of-ownership, the identified maintenance cost improvement attributable to ARCS could not be creditably quantified within the scope of this study. To evaluate the cost effects of modified maintenance procedures, a restructured maintenance facilities network, and projected occurrences and durations of delays, would require a highly sophisticated operational simulation of the selected airline model. Since such a simulation was beyond the scope of the ARCS program, these cost-effect factors have been identified and discussed strictly subjectively.

6.2.1.2 Cost of Weather Interruptions

In assessing the number of weather-caused diversions potentially avoidable by using the ARCS, consideration was given to three parameters: the number of scheduled arrivals into airports with Category II or III ground facilities, the probability of encountering Category II or lower weather conditions at these facilities, and the probability that the flight control system is functionally capable of performing the autoland task at the time it is required. This relationship can be expressed as follows:

$$\text{Avoidable diversions} = \Sigma (\text{no. of flights to Cat II runway}) \cdot P(\text{Cat II}) \cdot A(\text{Cat II}) + \Sigma (\text{no. of flights to Cat III runway}) \cdot P(\text{Cat III}) \cdot A(\text{Cat III})$$

where A = availability.

Table 16 summarizes the airports which, according to FAA planning, are scheduled to include Cat II and/or Cat III ground facilities by 1980. Weather probabilities were extracted from United Airlines weather department records for the last 14 years. This information shows that, on a probabilistic basis, the fleet of aircraft defined by the airline operational model will incur Category II weather conditions on 957 arrivals and Category III conditions on 1022 arrivals annually.

Using the defined airline operational model (providing for maintenance capability at 21 of its 77 destinations) and an average flight duration of 1.3 hours, and considering hardware and functional reliabilities, the average availability is 0.999 for the ARCS and 0.964 for the WWCS. Based on these availability parameters, the ARCS-equipped airplanes can be expected to incur approximately 33 fewer Category II and 36 fewer Category III weather-caused

Table 16.—Weather-Caused Interruptions

Category II Airports				Category III Airports			
Airport	P (Cat II)	Flights/day	Diversions/ year	Airport	P (Cat III)	Flights/day	Diversions/ year
ATL	0.012	15	65.7	ATL	0.009	15	49.28
BAL	0.005	9	16.43				
BDL	0.006	2	4.38				
BHM	0.001	4	1.46				
BOS	0.006	4	8.76				
BUF	0.003	7	7.67				
CLE	0.004	35	51.10				
CLT	0.008	3	8.76				
DAY	0.004	3	4.38				
DCA	0.003	18	19.71				
DEN	0.002	41	29.93				
DTW	0.002	6	4.38	DTW	0.004	6	8.76
EWR	0.003	4	4.38				
GEG	0.008	6	17.52				
HSV	0.001	6	2.19				
IAD	0.008	5	14.60	IAD	0.012	5	21.90
JFK	0.003	3	3.29	JFK	0.011	3	12.05
LAX	0.005	43	78.48	LAX	0.019	43	298.21
LGA	0.007	12	30.66				
MCI	0.002	7	5.11	MCI	0.002	7	5.11
MEM	0.002	4	2.92				
MKE	0.007	5	12.78				
MSP	0.002	12	8.76				
MSY	0.005	2	3.65				
OAK	0.002	6	4.38				
ONT	—	4	0				
ORD	0.008	97	283.24	ORD	0.005	97	177.03
PDX	0.005	23	41.98	PDX	0.018	23	151.11
PHL	0.005	11	20.08				
PIT	0.004	22	32.12				
PVD	0.008	2	5.84				
RIC	0.005	2	3.65				
ROC	0.003	10	10.95				
SEA	0.005	22	40.15	SEA	0.021	22	168.63
SFO	0.003	51	55.85	SFO	0.007	51	130.31
SLC	0.001	11	4.02				
SMF	0.007	6	15.33				
TOL	0.004	5	7.30				
TPA	0.004	2	2.92				
TYS	0.010	6	21.9				
Total			956.71			Total	1022.39

diversions than the WWCS-equipped airplanes. At \$2100 per diversion (an average amount based on United's fleet experience), this amounts to an annual savings of \$145 000.

6.2.2 ACQUISITION COST

The first cost advantage of the ARCS, compared to the WWCS, is in the acquisition cost. A pricing exercise, based on a production run of 990 units (300 ship sets plus 10% spares) for both the ARCS and an advanced WWCS (i.e., the WWCS architecture with current piece-part technology), yielded the following unit costs:

ARCS computer	\$30 500/unit = \$ 91 500/ship set
WWCS	
• Computer unit	\$21 000/unit
• Interface unit	\$21 000/unit
	\$42 000/system = \$126 000/ship set

The cost difference between the WWCS and ARCS is attributed to two factors: (1) the additional hardware required to implement triple-modular-redundant system voters, monitors, and other triplex-oriented special features and (2) the two-box nature of the WWCS. The cost contribution of these two factors is estimated to be approximately equal.

The specified cost is for hardware only. Estimated software development costs are \$1 million for the ARCS and \$850 000 for the WWCS. Amortized over 990 units, this results in an incremental unit cost of approximately \$1000 and \$850, respectively.

6.2.3 SYSTEM TEST COST EFFECT

This section discusses the analysis of economic benefits attributable to an integrated system test capability. Though much operating cost data is currently available for contemporary analog flight control systems, comparisons between such systems and the digital configuration represented by the ARCS are not considered to be valid. The purpose of the analysis therefore was to determine the cost effectiveness of the implemented system test function on its own merits. Maintenance of the ARCS hardware with conventional methods was compared to maintenance of the same ARCS hardware using the built-in system test capability. The intermediate-term application model with a flight-mandated (required by operator for dispatch) Category III autoland capability was assumed for this analysis.

The costs associated with the daily operation of an avionic system are either directly or indirectly incurred as a result of the maintenance operation and are therefore functions of the system configuration and the airline maintenance process. They are comprised primarily of line maintenance costs, shop maintenance costs, sparing expenses, and delay and cancellation costs attributable to the particular system. Line maintenance expenses are primarily man-hour labor costs and the associated overhead. There is, however, the potential for a delay cost if the maintenance activity takes excessive time.

Shop maintenance costs accumulate from several sources: the direct man-hour labor associated with testing and repairing equipment that has been removed from the aircraft; the material expense for repairing faulted units; overhead expense incurred as a function of direct labor; and automated test equipment (ATE) operating costs. System test is not expected to influence either material or overhead expenses. It will, however, have a beneficial effect on labor and ATE test time.

Sparing expenses accumulate from two sources: the cost of acquiring required spare units and the cost of holding the spares in inventory. The acquisition cost of spares is considered a variable rather than fixed expense because the level of sparing is a function of the demand rate and the demand rate is in turn a function of the maintenance operation. Inventory cost is generally computed as a percentage of the acquisition cost of the units being spared. Therefore, it is also a function of the maintenance operation.

Where it is necessary to deal with specific sensor systems, those assumed for the intermediate-term application model were used. However, when such parameters as sensor removal rates need to be defined, experience with the latest generation of avionic equipment was used as a basis for intermediate-term model projections. The latest generation is that represented by current DC-10 and 747 fleet operations.

The direct hourly labor rate assumed for all calculations was \$8 per hour. Overhead rate for line and shop maintenance operations was computed as 185% of direct labor.

The failure rate for the ARCS computer unit was estimated to be 320 failures per million hours, equivalent to an MTBF of 3125 hours. These estimates were derived from piece-part reliability projections using component failure rates in accordance with MIL-HDBK-217B. Removal rates for ARCS computer units were assumed to be 1.05 times the failure rate with system test capability and 2.0 times the failure rate without. The former assumes achievement of the system test design goal of no more than 5% nuisance failure indications, and the latter is indicative of current verification rate experience with contemporary flight control systems, which ranges from 25% to 60%. Without built-in self-test capability, it is unlikely that a digital implementation of the flight control function will significantly alter the current experience. A comparison of verification rates for analog and digital air data computers supports this: the analog computer used on the 747 and the digital computer used on the DC-10 exhibit verification rates of 32% and 30%, respectively.

The following analyses discuss the five cost categories of line maintenance, shop maintenance, spares provisioning, sensor system effects, and system test acquisition cost in greater detail.

6.2.3.1 Line Maintenance Cost

Line maintenance action is initiated by the flightcrew generating a flight-log writeup of the malfunctioning system. The line maintenance operation must then locate the failure within the system and effect the necessary repairs. A typical line maintenance operation has been defined and illustrated in figure 68. It consists of isolating the failure to a specific LRU, removing the failed unit, replacing it with a spare from stock, and functionally checking the newly installed unit. If the repair affects Cat II/III equipment, a test is made to verify the basic functional integrity of the system.

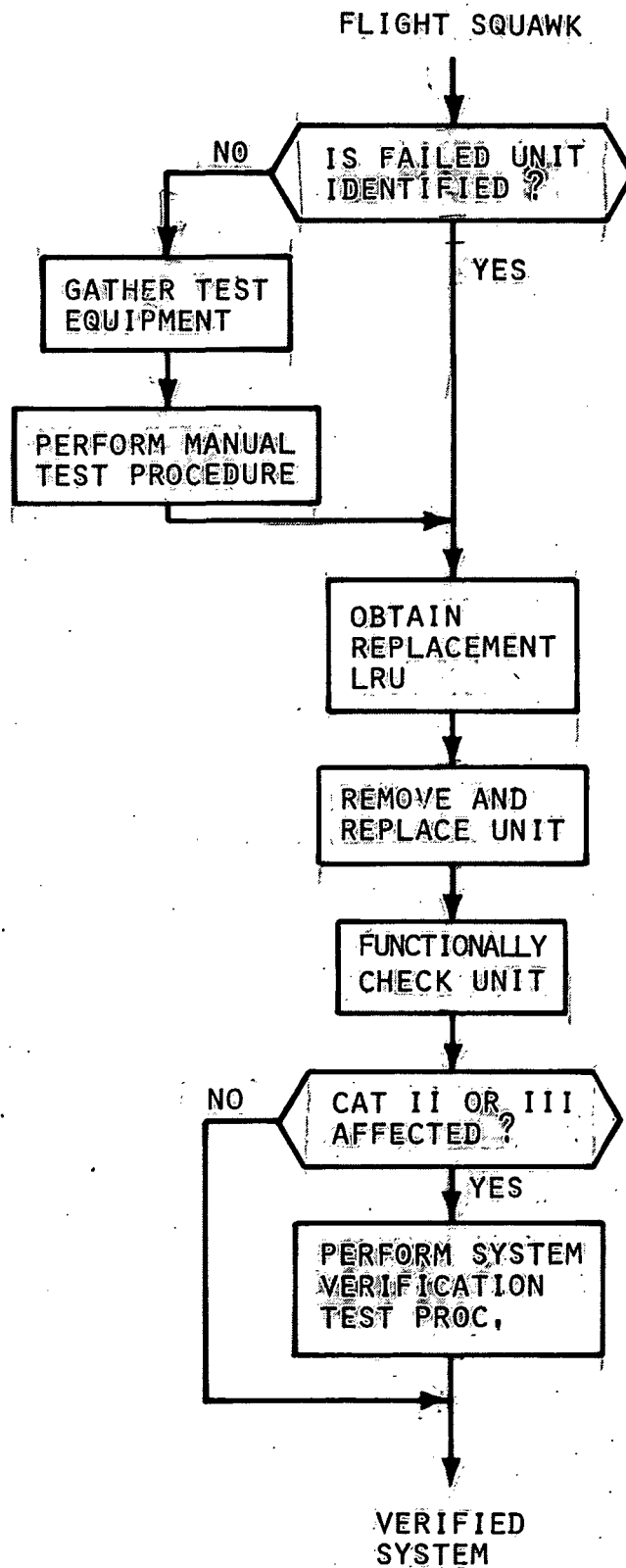


Figure 68.—Line Maintenance Operation

The only cost factors directly attributable to line maintenance are the incurred man-hours and associated overhead. Therefore, to evaluate the cost impact of this operation, it was necessary to make some time estimations for each of the activities involved. The time estimations for the ARCS with and without the system test capability are summarized in tables 17 and 18, respectively.

Table 17.—Line Maintenance Activity With System Test Capability

Activity	Time estimate, minutes
1. Deduce LRU location of fault	
• Read back in-flight fault recording	0.5
• Perform system BITE test	5
2. Remove and replace faulted LRU ^a	5
3. Functionally test new unit	4
4. Cat II or III verification test	5
5. Clean up	5
Total	24.5 minutes = 0.41 hour

^aExcludes time required to obtain replacement unit from stock, which can require from 30 minutes to 2 hours.

Table 18.—Line Maintenance Activity Without System Test Capability

Activity	Time estimate, minutes
1. Deduce LRU location of fault	
• Gather required test equipment	15
• Perform maintenance manual test procedure, or	60 ^a
• Assume from log entry	—
2. Remove and replace faulted LRU ^b	5
3. Functionally test new unit	5
4. Cat II or III verification test	60 ^a
5. Clean up	5
Total	150 man-minutes = 2.5 man-hours

^aRequires two maintenance technicians at 30 minutes each.

^bExcludes time required to obtain replacement unit from stock, which can require from 30 minutes to 2 hours.

One additional assumption was made with regard to the localization of a fault to a particular LRU. Half of the time the localization will be directly deduced from the nature of the squawk or, at most, only minimal testing will be performed. This assumption was based on past experience which shows that 50% of the incurred failures were adequately handled by "gate level" maintenance and that the other 50% required further testing. With this assumption, the average line maintenance man-hour requirement without system test capability is 2.50 hours when localization testing is performed and 1.25 hours when the fault location is assumed, or 1.88 hours per removal.

Computation of line maintenance cost becomes a function of the number of removals per year, the man-hours required, and the burdened hourly labor rate:

Unit flying hours = (3 units/airplane) (350 000 airplane hours/year) = 1.05×10^6 hours/year

with System Test							
Unit flying hours/year	x Failures/ 10^6 hours	x Removals/ failure	= Removals/ year	x Man-hours/ removal	x Cost/ hour	x Overhead factor	= Total cost
1.05×10^6	320	1.05	352.8	0.41	\$8	2.85	\$3297.97 or \$3300.00
without System Test							
1.05×10^6	320	2	672	1.88	\$8	2.85	\$28 804.61 or \$28 800

6.2.3.2 Shop Maintenance Costs

Once a unit is removed from the airplane, it becomes the responsibility of the shop maintenance operation to make the necessary repairs to restore the unit to an operable condition and return it for use. The shop maintenance operation can be viewed in three steps: first, the unit must be tested to locate the source of the failure; second, the failed elements must be repaired; and third, the unit must again be thoroughly tested to verify its fault-free operation. This operation is shown in figure 69.

Shop maintenance costs accumulate from several sources: direct man-hour labor; overhead computed as a function of direct labor; material expenses necessary for repairs; and the cost of using automatic test equipment when applicable. Since the cost comparisons of interest deal with the same basic ARCS hardware, it was assumed that there will be no difference in material expenditures, and hence the following computation considers only the time-related parameters.

The trend in current shop maintenance operations is to perform unit testing with the aid of automatic test equipment (ATE). ATE simplifies shop maintenance procedures but adds an additional hourly cost factor, estimated to be \$23. It was assumed that the ARCS would be tested using ATE in an airline maintenance program. That is, ATE will always be required

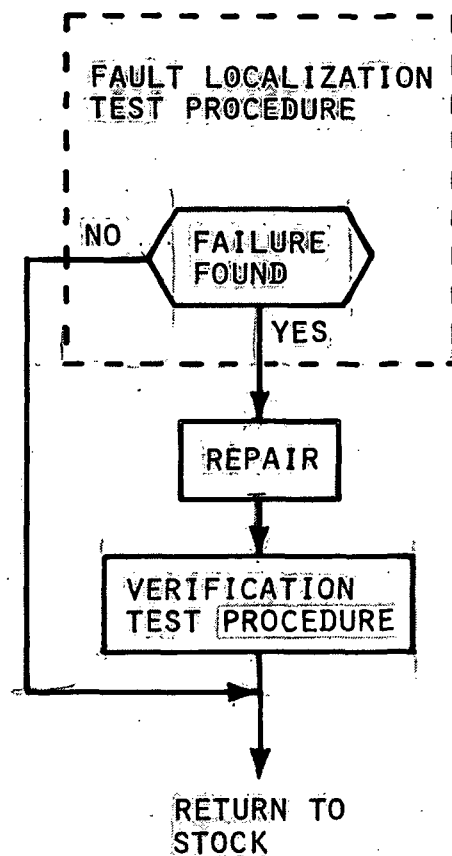


Figure 69.—Shop Maintenance Operation

to perform the postrepair verification test, even though operator experience has shown that when a sufficiently comprehensive built-in self-test is provided, it can often be used for fault localization in lieu of ATE. The major ARCS element, i.e., the computer LRU, will certainly include such capability as a part of the system test function, and it was assumed that this will adequately cover 95% of all incurred failures as defined by the system test coverage goal. The other 5% will require full use of ATE to localize the failure, and the ATE will be used to make a 100% verification test of the unit's operation.

A review of ATE running times for contemporary analog flight control systems indicates an average run time of 1.5 to 2.0 hours when no faults are found and when no operator intervention is required. The only digital system for which data is available is the air data computer for the DC-10, and it too requires 1.5 hours. The assumption was therefore made that the ARCS would require 1.5 hours of ATE time to run with no intervention and no failures. However, experience has shown that when ATE is being used for troubleshooting/fault-isolation purposes, the actual ATE time is about twice the minimum run time, or approximately 3 hours.

Another assumption made was that all unverified removals will require a full ATE checkout before they can be declared operational. The shop maintenance technician cannot know until after the full test that the removal was unjustified.

For the following calculations, a time profile for the shop maintenance cycle was derived and is shown in table 19. For the ARCS without system test, all incurred faults are "non-covered" faults. All unverified removals are treated the same with or without built-in system test.

Table 19.—Shop Maintenance Operation

Activity	Time estimate, hours		
	No fault	BIT-covered fault	Non-BIT-covered fault
1. Locate failed element			
• Built-in test (BIT) time	0.33	0.33	0.33
• ATE test time	3.0		3.0
2. Repair		1.0	1.0
3. ATE verification test		1.5	1.5
4. Miscellaneous	0.5	0.5	0.5
Total man-hours/ATE-hours	3.83/3.0	3.33/1.5	6.33/4.5

The shop maintenance costs are the sum of the costs associated with "covered" faults plus "noncovered" faults, plus the cost of unverified removals. These computations are as follows:

$$\text{Shop cost} = \Sigma (\text{number of occurrences}) [\text{man-hours/occurrence (cost/hour)} + \text{ATE-hours/occurrence (ATE cost/hour)}]$$

with System Test

Event	No. of occurrences	Man-hours	x Cost/hour	+ ATE-hours	x ATE cost/hour	= Total cost
BIT-covered fault	336 x 95%	3.33	\$22.80	1.5	\$23	\$35 247.34
Non-BIT-covered fault	336 x 5%	6.33	22.80	4.5	23	4 163.44
No fault	352.8 - 336	3.83	22.80	3.0	23	2 626.24
						\$42 037.02
						or
						\$42 000.00

without System Test						
Event	No. occurrences	Man-hours	x Cost/hour	+ ATE hours	x ATE cost/hour	= Total cost
Fault found	336	6.0	\$22.80	4.5	\$23	\$ 80 740.80
No fault found	672 - 336	3.5	22.80	3.0	23	49 996.80
						\$130 737.60
						or
						\$131 000.00

6.2.3.3 Spares Provisioning Costs

Because avionic equipment fails and it takes a finite amount of time to repair, an airline must maintain a sufficient quantity of spares to support continuous fleet operation. The costs associated with spares provisioning arise from two sources: the initial acquisition cost and inventory or holding costs. The quantity of spare LRU's required to support an operation is in part dependent on the line demand rate, a direct result of removal rates. Inventory or holding costs are normally computed as a percentage of initial acquisition cost.

Sufficient spare units must be available to satisfy the demand at each line maintenance station and to fill the repair cycle pipeline. The formula derived for this is as follows:

$$\text{Total spares} = \text{Base pool} + \text{in transit} + \text{line spares}$$

The base pool consists of at least one "on-hand" spare plus the average number of LRU's in the shop repair cycle, which is the product of the average daily usage times the average cycle time. In transit simply says that it takes a certain amount of time to transport a failed unit from the line station at which it was removed to the repair point. Line spares are allocated to the various line stations in accordance with the projected demand, but as a minimum, there will be one spare LRU supplied to each line maintenance station. For this analysis, one spare LRU per station was assumed. For the operational models, these factors were:

$$\begin{aligned} \text{Base pool} &= 1 + (\text{average daily usage}) (5\text{-day cycle time}) \\ \text{In transit} &= (\text{average daily usage}) (3\text{-day transit time}) \\ \text{Line spares} &= (1 \text{ per line station}) (21 \text{ line stations}) \end{aligned}$$

With System Test

$$\begin{aligned} \text{Total spares} &= 1 + 5 \left(\frac{352.8 \text{ removals/year}}{365 \text{ days/year}} \right) + 3 \left(\frac{352.8 \text{ removals/year}}{365 \text{ days/year}} \right) + 21 \\ &= 30 \text{ units} \end{aligned}$$

Without System Test

$$\begin{aligned} \text{Total spares} &= 1 + 5 \left(\frac{672 \text{ removals/year}}{365 \text{ days/year}} \right) + 3 \left(\frac{672 \text{ removals/year}}{365 \text{ days/year}} \right) + 21 \\ &= 37 \text{ units} \end{aligned}$$

To derive the total cost difference attributable to initial spares provisioning, it is necessary to know the acquisition cost per unit. The ARCS computer LRU's were estimated to cost approximately \$30 500 per unit in production quantities. Therefore, the cost of spare computer LRU's was:

$$\begin{aligned} \text{With system test} &= (30 \text{ units}) (\$30\,500/\text{unit}) = \$915\,000 \\ &= \frac{\$915\,000}{16 \text{ years}} = \$57\,190/\text{year} \end{aligned}$$

$$\begin{aligned} \text{Without system test} &= (37 \text{ units}) (\$30\,500/\text{unit}) = \$1\,128\,500 \\ &= \frac{\$1\,128\,500}{16 \text{ years}} = \$70\,500/\text{year} \end{aligned}$$

Experience shows that the annual cost for holding spare LRU's is equal to approximately 25% of their purchase cost. Therefore, spares holding costs from the above purchase cost are:

$$\begin{aligned} \text{With system test} &= (0.25) (\$915\,000) = \$228\,800 \\ \text{Without system test} &= (0.25) (\$1\,128\,500) = \$282\,100 \end{aligned}$$

Therefore, the computed annual cost of providing spare computer elements for the ARCS flight control system is:

	<u>With system test</u>	<u>Without system test</u>
Amortized acquisition cost	\$57 000	\$70 500
Spares inventory cost	\$228 800	\$282 100
Total	\$286 000	\$353 000

6.2.3.4 Sensor System Maintenance Costs

The AFCS is a complex system involving much more than just control computers. Its operational success is contingent upon proper operation of all of the sensor and servo systems associated with it. A failure occurrence in any one of the peripheral sensor or servo systems can potentially result in a degradation of AFCS performance and hence initiate a maintenance action. Experience with contemporary flight control systems has shown that the normal corrective action for most such squawks is to replace the major element—the flight control computer. This practice is reflected in the low verification rates currently achieved for these units. This effect has been taken into account in the cost analysis by the assumption of a 50% unverified removal rate for ARCS computers without system test capability. However, the reverse situation is also present, though to a lesser degree; sensor systems are erroneously removed and replaced as a result of a flight control problem. One important attribute of the system test function is the capability to detect and register the occurrence of such sensor failures to minimize unjustified removals.

C-3

A review of in-use experience with contemporary sensor systems on DC-10 and 747 fleets indicates that approximately 7% of all removals of air data computers, ILS receivers, and radio altimeters were initiated as a result of a squawk against the flight control system. Of these 7%, approximately 60% proved to be unverified. With the achievement of the system test goal of a 90% effectiveness in properly isolating sensor system failures, it can be projected that the percentage of sensor removals due to flight control squawks could be reduced to 3.1%. This reduction in sensor removals will also have an effect on the system test cost effectiveness.

Table 20 defines 1975 average removal rates for 747 and DC-10 avionic equipment, adjusted to a fleet size indicative of the selected operational model. If we assume that without system test the sensor removal rate will remain the same as for contemporary systems, we could expect a saving of 3.9% of the total projected removals, equal to 36 unwarranted ILS receiver removals per year, 34 radio altimeter removals, and 47 air data computer removals.

Table 20.—Sensor System Removals

Equipment	Removals per year	Adjustment factor	Projected annual removals	Annual AFCS-related removals	
				With system test ^a	Without system test ^b
ILS receiver	228	$\frac{1\ 050\ 000}{260\ 000}$	921	28.65	64.47
Radio altimeter	216	$\frac{1\ 050\ 000}{260\ 000}$	872	27.13	61.04
Air data computer	300	$\frac{1\ 050\ 000}{260\ 000}$	1212	37.71	84.84

^aBased on 3.111% removals due to flight control squawks

^bBased on 7% removals due to flight control squawks

Table 21 identifies parameters pertinent to the computation of sensor maintenance costs. Both line and shop maintenance time requirements have been obtained from current operating experience and represent an average of 747 and DC-10 data.

Table 21.—Sensor Maintenance Parameters

Unit	Cost/unit	Line maintenance, man-hours per removal	Shop maintenance, man-hours per removal
ILS receiver	\$ 4 800	1.27	10.25
Radio altimeter	\$ 5 550	1.29	4.21
Air data computer	\$18 070	1.14	7.81

Using these parameters, the following cost calculations show sensor maintenance effects on system test cost-of-ownership.

Line maintenance cost = Σ (no. of removals) (man-hours/removal) (cost/man-hour)

With System Test

ILS:	28.65 (1.27 hours) (\$22.80/hour)	=	\$ 829.59
Radio altimeter:	27.13 (1.29 hours) (\$22.80/hour)	=	797.95
Air data computer:	37.71 (1.14 hours) (\$22.80/hour)	=	980.16
Total			\$2607.70
		or	\$2600.00

Without System Test

ILS:	64.47 (1.27 hours) (\$22.80/hour)	=	\$1866.79
Radio altimeter:	61.04 (1.29 hours) (\$22.80/hour)	=	1795.31
Air data computer:	84.84 (1.14 hours) (\$22.80/hour)	=	2205.16
Total			\$5867.26
		or	\$5900.00

Shop maintenance cost = Σ (no. of removals) (man-hours/removal) (cost/man-hour)

With System Test

ILS:	28.65 (10.25 hours) (\$22.80/hour)	=	\$6695.51
Radio altimeter:	27.13 (4.21 hours) (\$22.80/hour)	=	2604.15
Air data computer:	37.71 (7.81 hours) (\$22.80/hour)	=	6714.94
Total			\$16 014.60
		or	\$16 000.00

Without System Test

ILS:	64.47 (10.25 hours) (\$22.80/hour)	=	\$15 066.64
Radio altimeter:	61.04 (4.21 hours) (\$22.80/hour)	=	5 859.11
Air data computer:	84.84 (7.81 hours) (\$22.80/hour)	=	15 107.29
Total			\$36 033.04
		or	\$36 000.00

Spares provisioning cost* = $\Sigma \frac{1}{16}$ (Acquisition cost) + inventory cost

*To cover spares required for 8 days' supply: 5 days' repair cycle time, 3 days' transit time.

With System Test

ILS:	$8 \left(\frac{28.65}{365} \right) = 0.628 \text{ units at } \4800 ea	=	\$ 3 014.14
Radio altimeter:	$8 \left(\frac{27.13}{365} \right) = 0.595 \text{ units at } \5500 ea	=	3 270.47
Air data computer:	$8 \left(\frac{37.71}{365} \right) = 0.827 \text{ units at } \$18 070 \text{ ea}$	=	<u>14 935.23</u>
Total acquisition cost			\$21 219.84

Annual acquisition cost = (1/16 x Total)	=	\$1326.24
Inventory cost = 0.25 (total acquisition cost)	=	<u>5304.96</u>
Total annual sensor spares cost	=	\$6631.20
	or	\$6600.00

Without System Test

ILS:	$8 \left(\frac{64.47}{365} \right) = 1.41 \text{ units at } \4800 ea	=	\$ 6 783
Radio altimeter:	$8 \left(\frac{61.04}{365} \right) = 1.34 \text{ units at } \5500 ea	=	7 358
Air data computer:	$8 \left(\frac{84.84}{365} \right) = 1.86 \text{ units at } \$18 070 \text{ ea}$	=	<u>33 601</u>
Total acquisition cost			\$47 742

Annual acquisition cost = (1/16 x total)	=	\$ 2 984
Inventory cost = 0.25 (total acquisition cost)	=	<u>11 935</u>
Total annual sensor spares cost	=	\$14 919
	or	\$15 000

6.2.3.5 Acquisition Costs

Acquisition costs attributable to the ARCS system test function are primarily associated with the system test panel (STP) and its associated interface electronics and the system test software development. The estimated acquisition cost attributable to the STP and its electronics was estimated to be \$6000 based on a production run of 330 units. System test software development costs were estimated to be 10% of the total ARCS software cost, which is approximately \$1 million.

Amortized over the 16-year operational life of the equipment, these two acquisition costs yield an annual expense of approximately \$433 per airplane, or \$65 000 for the defined fleet.

6.2.3.6 System Test Cost Conclusions

Table 22 summarizes the potential improvement in annual direct maintenance costs that can be expected from the inclusion of an integrated system test function. These cost factors are expressed as annual costs for a fleet of 150 aircraft in the defined operating environment. Taking the maintenance cost saving and subtracting the acquisition cost of system test yields an annual saving of \$1000 per airplane per year, or \$16 000 over the 16-year operational lifetime of each airplane.

Table 22.—Maintenance Costs Per Year

Cost factor	With system test	Without system test	Difference
Line maintenance			
Computer	\$ 3 300	\$ 28 800	\$ 25 500
Sensor systems	2 600	5 900	3 300
Shop maintenance			
Computer	42 000	131 000	89 000
Sensor systems	16 000	36 000	20 000
Spares provisioning			
Computer	286 000	353 000	67 000
Sensor	6 600	15 000	8 400
Total	\$356 500	\$569 700	\$213 200

6.2.4 COST/BENEFIT SUMMARY

ARCS showed an acquisition cost saving of approximately \$34 000 per ship set compared to the WWCS. This cost saving is attributable to the ARCS computer and I/O being placed in a single unit, as well as to the ARCS redundancy management functions being implemented primarily in software.

In assessing the number of weather-caused diversions that could be avoided because of the ARCS greater availability, three parameters were considered: the number of scheduled arrivals at airports with Cat II or Cat III ground facilities, the probability of encountering Cat II or Cat III weather conditions, and the probability that the required flight control system function was available. Using the defined airline model and a specific route structure typical of United's 727 fleet, the ARCS-equipped airline can be expected to incur approximately 69 fewer diversions per year as compared to the WWCS. Based on an average United Airlines cost per diversion, this can save approximately \$145 000 annually.

Including the system test maintenance feature in the ARCS can further save approximately \$150 000 for the same airline model. This cost saving is brought about largely by improving the effectiveness (success probability) of the maintenance action, thereby reducing the number of unwarranted equipment removals experienced with today's systems.

The cumulative effect of system acquisition cost and avoided diversion expense for the defined airline model is an annual cost saving of approximately \$495 000, more than \$3000 per aircraft. The inclusion of system test yields a further saving of approximately \$1000 per aircraft per year.

Other cost savings attributable to the ARCS can be expected as we look deeper into the cost-of-ownership picture—reduced spares, maintenance procedural improvements, dispatch availability, etc.

7.0 ARCS IMPLEMENTATION

The ARCS study was the first phase of a potential three-phase NASA program to specify, design, implement, and test an advanced airborne reconfigurable computer system. In anticipation of a second phase—the ARCS hardware implementation—the question had to be considered whether the best alternative would be to modify existing NASA equipment, to adapt other suitable available hardware, or to build entirely new hardware. Included in the ARCS work statement, therefore, were the tasks of assessing the feasibility of modifying the GE MCP-703 (WWCS) to conform to the ARCS configuration and of identifying suitable alternate, commercially available airborne digital systems adaptable to the ARCS design.

The following sections present the computer system design specification resulting from the design synthesis and analysis efforts of the ARCS study (sec. 7.1) and the assessment of the feasibility of modifying the WWCS to the ARCS configuration (sec. 7.2).

7.1 ARCS DESIGN SPECIFICATION

This section specifies the functional, software, and hardware design requirements applicable to an advanced reconfigurable computer system. The specification will serve as a basis for evaluating the fault-tolerance capability of existing airborne computer systems intended for application to commercial transports and for establishing the degree of modification required, if any, to upgrade such systems to ARCS standards.

The design specification is divided into five sections. The first section covers the real-time aspects of the system, the second and third sections cover the ground test and on-line monitoring aspects, respectively, and the fourth section covers the fault-tolerant aspects of the system in detail. Functional, software, and hardware considerations are integrated into each of these sections. The last section covers hardware design requirements of a fault-tolerant computer system.

7.1.1 ARCS REAL-TIME OPERATIONS

To ensure proper execution control of ARCS critical real-time operations, the software design shall be functionally organized into modules that use only the basic single-entry/single-exit constructs of concatenation, loop, and if-then-else. The organization of the software shall also reflect the data requirements of each system module and result in a hierarchical data structure with minimum intermodule data requirements.

The real-time operations shall be functionally structured into synchronous operations, asynchronous operations, and interrupt operations. Synchronous and interrupt operations are further specified in the next two sections. Asynchronous operations are covered in section 7.1.3.

7.1.1.1 Synchronous Operations

The time-critical application functions (control law computations and mode control logic) and redundancy management functions shall be processed in frame-time

synchronization. Synchronous operation is preferred to asynchronous to: (1) facilitate bit-similar/bit-identical processing to eliminate the need for feedback equalization, (2) allow timely fault localization/isolation and recovery initiation by minimizing processing wait times insofar as cross-channel information exchanges are concerned, (3) allow tighter fault detection thresholds, and (4) minimize cross-channel labeling and associated RAM requirements.

Frame synchronization shall be performed by a software algorithm based on cross-channel exchange of sync status information. The processing and interface requirements for the application functions are specified in table 23. Redundancy management functions are specified in section 7.1.4.

Table 23.—Processing and Interface Requirements for ARCS

Required capability	Near-term application	Intermediate-term application	Far-term application
Instruction throughput, kops (85% add, 10% multiply, 5% divide)	300	400	600
Minor frame time, ms	20	20	10
ROM size, K	8	8	8
RAM size, K	4	4	4
Analog inputs	24	16	42
Discrete inputs	24	16	12
Digital inputs	3	3	4
Analog outputs	8	5	14
Discrete outputs	20	18	18
Digital outputs	1	1	2

7.1.1.2 Interrupt Operations

All processing control functions initiated by a hardware interrupt shall be functionally organized into one software module. Hardware interrupt functions shall be provided for the following purposes:

- Frame-time iteration reference
- Arithmetic overflow
- Memory parity fault
- System test panel input
- Power-on

All interrupts except power-on shall be maskable in software.

7.1.2 GROUND TEST OPERATIONS

The ARCS automated built-in test and diagnostic function shall test for failures within the ARCS (sensors, computers, and servoactuators) and localize them to a line replaceable unit (LRU). The ground testing function shall be available on the ground only when requested by the operator.

Ground test shall use a "center out" test philosophy, wherein basic elements are tested first and then used to test other functions (computer LRU followed by sensors and concluding with servo systems). The computer testing shall include central processor, hardware monitors, RAM and ROM memories, and input/output elements.

7.1.2.1 Processor Testing

The processor self-test shall perform an instruction test sequence within which all instructions or, as a minimum, all microinstructions are exercised, all registers involved, and all addressing modes used.

7.1.2.2 Hardware Monitors

The ability of each of the computer hardware monitors to detect and enunciate the existence of a fault condition shall be verified as part of the ground test.

The ability of the watchdog monitor to indicate a "good" state and a failed state, and the ability of the arithmetic error detector to detect an overflow condition and generate the corresponding interrupt, shall be verified.

A parity generator inversion discrete, controllable by software, shall cause the words to be stored in memory with even parity to provide a test of the operation of the parity error interrupt. Where separate parity error detectors are used for different sections of memory, this test shall be provided for each parity detector in the system.

7.1.2.3 Memory Testing

Read-only memory (ROM) shall be tested using sum checking techniques. Random access memory (RAM) shall be checked by writing a predetermined data pattern into all accessible RAM and comparing the readout.

7.1.2.4 Input/Output Testing

The hardware shall include the capability to switch, under software control, all output signals into all input channels.

7.1.2.5 Sensor Testing

Provisions shall be made for verifying the operational integrity of the sensors and their associated interfaces. Sensor testing shall make use of sensor system self-testing capabilities where available.

7.1.2.6 Servo Testing

Servo subsystems shall be tested for satisfactory engagement/disengagement control, dynamic response, and force override characteristics. The servo ground test function shall be initiated only after a specific request from the test operator.

7.1.2.7 Ground Test/Crew Interface

The interface between the flight or ground crew and the system test function is the system test panel (STP), which shall provide the operator with the means to initiate the test function and display the test results.

7.1.3 ASYNCHRONOUS OPERATIONS

Asynchronous operations are those tasks that need not be performed simultaneously in all computers. The asynchronous operations shall include on-line self-testing, maintenance data update, and STP processing.

On-line self-testing shall detect and/or localize failure conditions not detectable by first-level system monitors. It shall include processor self-test/diagnostics, memory testing, and input and output electronics testing. Input and output testing capability shall be provided by a continuous wrap-around loop test utilizing dedicated output and input channels.

The maintenance data update task shall centrally collect all failure data for maintenance purposes, including failure information accumulated during the previous in-flight test and the failure status of the system monitors, i.e., SSFD, computer output monitor, and servo monitor. The maintenance data update process shall resolve the LRU location in which a failure has occurred and, if not previously done, record it in a nonvolatile section of memory for future use by maintenance personnel.

The STP processing shall interface the on-line test program with the operator. It shall accept operator request inputs, as well as process and format output data to be displayed to the operator.

7.1.4 REDUNDANCY MANAGEMENT

Redundancy management functions shall be structured into four groups: cross-channel synchronization, cross-channel data transmission, reconfiguration, and recovery update.

Each of these functions shall be performed so that each redundant channel can autonomously assess the redundancy status of the system and, based on this assessment, operate in the corresponding channel redundancy state. Under no circumstances shall one computer operation, or combination of computer operations, interrupt the normal operation of another computer.

7.1.4.1 Cross-Channel Synchronization

The synchronization process following recovery from a loss-of-sync fault shall be identical to that following power-on. Initial synchronization and normal frame synchronization shall be performed by system software via setting and clearing of sync indicators that are cross-channel exchanged between each pair of computers.

7.1.4.2 Cross-Channel Data Transmission

All variable data that are input, output, or history of the computation during each minor frame shall be spontaneously cross-channel exchanged between all computers. The receiving computer has sole jurisdiction in the use of the data.

The data rates required for cross-channel transmission of variable data for the ARCS applications are:

<u>Near term</u>	<u>Intermediate term</u>	<u>Far term</u>
20K	25K	50K

7.1.4.3 Reconfiguration

Reconfiguration includes fault detection, fault localization, and fault isolation of sensor, computer, and servo functions.

Sensor signal selection and failure detection shall be performed by software using cross-channel sensor data. The SSFD algorithm shall meet the following requirements.

- The sensor selection algorithm must be able to isolate the effects of all types of faults, including open failures, hardover failures, and any ramp failures and oscillatory failures.
- The signal selection algorithm must provide an output that is acceptable to the application task for normal (unfailed) operation, and during any fault condition in one of the redundant input signals.
- Means must exist to detect and isolate the effect of a sensor failure, and to revise the failure-detection algorithm to be compatible with the lower redundancy, before the probability of incurring an additional like failure may be high enough to create an unacceptable risk.
- The failure-detection algorithm must operate in the presence of normal signal tolerances, such as biases, scale factors, and linearity errors, and in the presence of noise.
- The proportion of "nuisance failures" (leaky transients) due to signal tolerances and noise must be insignificant compared to the number of genuine failures.

Fault monitoring shall be performed by software on computational results prior to output of servo commands. The output monitor shall establish the validity of the local computer's processing based on comparisons with other computer's results. If the output monitor determines that a local computer fault has occurred, it shall initiate disengagement of all affected servos.

A software monitor shall check the integrity of the cross-channel data buses.

Recovery of a computer, as indicated by reestablished synchronization, shall cause working computers to reset their failure status indicators. Recovery for a faulted computer shall consist of updating its failure status indicators and variable data history to a working computer.

Pilot intervention shall not be relied upon for system reconfiguration processes. The system shall be designed with the capability of automatic start and synchronization after power turn-on and automatic restart and resynchronization after transient power faults or massive transient signal faults.

7.1.5 HARDWARE DESIGN REQUIREMENTS

To support assumptions on which the ARCS design analyses were founded, and which subsequently resulted in the selected ARCS configuration, the following specific hardware design requirements shall apply.

7.1.5.1 Independent Computer Monitor

An independent, fail-safe monitor of the real-time operation of each computer shall be an integral part of each computer unit. If the measured time interval between the clearing of consecutive sync indicators is not within specified upper and lower limits of the nominal frame time period, the independent monitor shall indicate a computer fault condition. If the computer's sync indications return to within the required periodic interval, the independent monitor shall clear the fault indication.

7.2.5.2 Channel Separation and LRU Packaging

Computer and interface hardware shall be dedicated on a per-channel basis. Based on maintenance and logistics considerations, a single LRU for computer and interface electronics is preferred.

The hardware design shall be constructed to survive a cable failure as a single failure, where all the wires in a cable may be open-end or shorted to ground. It shall also survive, as a single failure, the computer LRU sliding out of the receiving tray, where all pins on the LRU connectors disengage almost simultaneously.

7.1.5.3 Design Integrity

The hardware shall be designed to withstand and suppress the effects of electrical hazards including lightning stroke. Components in separate channels shall be completely isolated electrically. Cross-channel data transfers between computers shall use serial, optical data links.

7.1.5.4 General Architecture

The computer architecture shall be compatible with both ROM/RAM and core memory. The hardware design shall be adaptable to minimum quadruplex redundancy. A basic word length of 16 bits with the capability of efficient double precision, or a basic word length of 24 bits, is required.

7.1.5.5 General Hardware Design Specification References

The hardware design shall meet the requirements specified by reference 3.

The components selected for the hardware design shall equal or exceed the following standards.

Microelectronics	B-1	Mil-STD-883, Method 5004, B
Discrete semiconductors	JAN	Mil-M-38510, C
Capacitors--Aluminum (ER)	R	
--All others (ER)	M	
Resistors (ER)	M	
All other (non-ER) parts	Mil-Spec	

7.2 FEASIBILITY OF WWCS MODIFICATION

When the ARCS program first got under way, it was thought that one desirable method of obtaining some actual hardware experience with many of the important ARCS architectural concepts would be to modify the WWCS. The WWCS was to serve as the point of departure and baseline system architecture for the ARCS study program. As a result, a task was defined for the ARCS contractors to delineate those modifications that could be made to the WWCS to bring it to a near-ARCS configuration at some future time.

Since the ARCS study program began, the electronics industry has made enormous strides in larger scale integration and lower power circuits. ARCS has taken advantage of these developments in circuit technology and has moved nearly all the redundancy management functions into software. Thus, the ARCS configuration represents a dramatically different architecture than the WWCS.

This section presents a series of modifications that could be implemented in the WWCS to produce a near-ARCS structure for evaluating many of the ARCS concepts. Although these modifications would permit experimentation with many of the ARCS architectural concepts, they result in a considerable decrease in available computational time for control laws within the WWCS. Considering control law experimentation computer system, these modifications are not recommended.

7.2.1 WWCS-ARCS COMPARISON

The primary differences between the ARCS and WWCS are the following.

1. The WWCS is a TMR system designed around a combination of hardware and software redundancy management, with the hardware playing a very strong role. The ARCS is a triplex system designed to degrade in the face of multiple failures to simplex operation for any module. The redundancy management is primarily in software, although the independent servo monitor and watchdog monitor are hardware functions.
2. The WWCS employs a fail-operational oscillator system to provide a bit-synchronous source of timing for input and exchanges and to generate the time base for real-time operations. The ARCS channels perform operations using completely independent timing sources and use interchannel discretes for frame synchronization.
3. The WWCS uses relatively restricted computer-controlled interchannel data links only for preflight test data exchange. The ARCS contains fast and flexible computer-controlled interchannel data links that employ a LIFO buffer in the transmitter and DMA storage of the received data into dedicated zones in a multiport memory.
4. The ARCS contains an independent watchdog monitor configured to detect when the processor is no longer a logical element and to disengage all the servoactuators connected to that channel. The WWCS contains no such circuit since there is no attempt to operate a single channel of the WWCS by itself as a degraded system mode.

7.2.2 WWCS MODIFICATIONS

The following modifications could be made without severely impacting the WWCS hardware.

1. Six cables would be rerouted to defeat the bit-synchronous timing throughout the system and permit each computer unit/interface unit pair to operate independently. The interchannel exchange of raw sensor data would be severed, and therefore, the utility of the hardware sensor select electronics would be deleted, which means that sensor redundancy management must be performed in software. This change must be accompanied by item 2 if the servo transmit/receive unit (STRU) is to remain a part of the system.
2. The interchannel cable would be modified to delete the interchannel exchange of STRU data and timing. This modification would permit each channel of the STRU to operate independently in conjunction with the corresponding computer and interface units.
3. The synchronous logic interface (SLI) electronics would be redesigned and reprogrammed to minimize the number of memory cycles used for input/output during each computational minor cycle consistent with interfacing with a single interface unit.

4. The status register would be extended by two bits to accommodate interchannel synchronization discretes and these discretes would be wired across channel. This would give each processor means for signaling all other processors for synchronization without using the processor-controlled input/output system, which is restricted to background usage in the MCP-703 system. This expansion of the status register could be accommodated on the SLI circuit boards.
5. An improved interchannel data link would be designed and installed with a conventional DMA structure on each end (transmitter and receiver) or with a FIFO or LIFO buffer on the transmitter end if a comparable block of memory is given up. This improved interchannel link could reside in the spare card locations that have been opened up through the provisions of an external memory.
6. A watchdog monitor comparable to that designed for the ARCS would be installed in the CIU in one of the spare card locations.
7. The processor would be given access to the error reset discrete on a full-time basis.
8. Substantial software modifications required for the above hardware modifications include the following.
 - a. A frame synchronization routine must be developed and integrated into the WWCS executive.
 - b. Recovery procedures must be finalized and the code required must be integrated into the WWCS software.
 - c. An extensive computer self-test routine must be developed.
 - d. A new BIT program must be written and integrated with the WWCS software package.
 - e. A new interchannel driver routine must be developed.
 - f. The sensor redundancy management routine must be modified to account for the different location of interchannel raw sensor data.
 - g. A new system test routine must be developed.

Although many other areas of the software would be affected in a minor way, these listed programs would be new or severely impacted.

8.0 CONCLUDING SECTION

This section summarizes the results of the ARCS study and presents the conclusions drawn from the results.

8.1 SUMMARY OF RESULTS

The results of the criteria development, the conceptual design, and the analysis tasks are restated below in summarized form. Complete information on these subjects resides in sections 4 through 6.

8.1.1 ARCS DESIGN CRITERIA

The design criteria derivation had the objective of identifying economic and flight safety considerations at the aircraft operational level that influence the specification of on-board computer systems. Specific design requirements and design principles for an airborne fault-tolerant (redundant) digital computer system were then formulated from an interpretation of the design criteria, applying flight-critical system design practices to avoid potential single-point system failures.

ARCS has the potential to reduce aircraft costs in three areas: the functional scope of the system, which influences the initial system cost as well as the potential operational benefits of the system; the functional availability of the system, which determines the degree of achieving the potential operational benefits; and the system maintainability, which influences the burden of maintaining the system at the functional status required to achieve the operational benefits.

A control/stability augmentation function and a Category III autoland function were postulated to define the scope of the ARCS.

The economics of an airplane's on-schedule performance dictates a particular function's desired or needed availability. This functional availability translates into computer system design requirements that will set the system's failure survival capability. No explicit functional availability criteria exist for the ARCS application functions. However, airlines have gone on record expressing a desired average probability for an operational Cat III capability. An average probability of having an operational Cat III capability greater than 0.9 with a 100-hour maintenance interval was postulated as a design goal for the ARCS design task.

Significant requirements identified for the third economic issue—system maintainability—were that only “condition monitoring” maintenance be acceptable, that functional integrity checks be automated, and that system test features be contained within the on-board equipment.

Flight safety criteria are an essential part in assessing the airworthiness of an airplane and its systems. Specifying safety-of-flight goals implies defining the level of accident risk, or probability of accident, that can be tolerated for the total aircraft, as well as the risk

contribution that can be tolerated because of particular subsystems in the aircraft. The basis for these requirements is established by the Federal Aviation Regulations. Interpretation of these regulations resulted in assessing the failure probability for the CCV/FBW function to be less than 1×10^{-9} for a 1-hour flight and failure probability for the autoland function to be less than 1×10^{-7} for a 45-second exposure.

Two fundamental design requirements for the fault-tolerant computer system were identified. First, the system must be capable of gracefully degrading from a triplex or quadruplex redundancy level to a simplex string of operable elements—sensors, computer, and servos. Second, the computer system must not be dependent on flightcrew intervention for startup or reconfiguration process initiation. The system must therefore be capable of automatically establishing fully redundant operation following power turn-on and must be able to automatically reestablish the highest operable level of redundancy following transient fault conditions.

Designing against a system single-point failure mode is absolutely necessary if the design is to meet the above operational requirements. Redundant channel interdependence is a potential source for such a failure mode; therefore, a channel independence was deemed to be an important requirement in the design of the reconfiguration processes. To provide design guidelines to achieve this independence, the following design principles were adopted: (1) each computer shall independently assess its own operational status; (2) no computer operation, or combination of computer operations, shall interrupt the normal operation of another computer; and (3) no servo shall be controlled by processes outside its own channel.

Reliability predictions, on which the operational risks during the use of the system are based, rest on assumptions about the failure status of the system at any particular time. The system failure status, therefore, must be established through a thorough verification process that immediately (or within a short time period) detects and localizes system failures. Further, the functional integrity of the system must be ensured after maintenance action has taken place. A system test function is therefore required to assemble the system failure data for the redundancy management process and to provide a means for checking the system functional integrity. The system test function is also to enhance system maintainability.

8.1.2 ARCS DESIGN CONCEPT

In the functional concept developed from the design requirements, the essence of the ARCS is the reconfiguration processes. In dealing with the various fault conditions that can cause a transition between the possible states of the computer subsystem, a single reconfiguration strategy was adopted. With this strategy, a power-up operation is identical to recovery from a transient power fault.

The general reconfiguration process involves one or more of the subprocesses of fault detection, fault localization, fault isolation, and recovery or redundancy degradation. Recovery from a transient fault will result in a restoration of the operational state that prevailed before the fault condition occurred. Failure of the recovery process, i.e., the fault is declared permanent, will result in a degradation in the redundancy state of the affected stage.

The heart of the ARCS is the redundancy management software. It is the implementation in code of this redundancy management function that brings together both the hardware and software aspects of the reconfiguration processes.

The ARCS concept emphasizes software processes in achieving fault-tolerant capabilities. The ARCS hardware must therefore be viewed as a vehicle to facilitate an effective software design for the overall reconfiguration and application processes.

The baseline ARCS configuration is a triplex system, readily expandable to quadruplex, with a single computer unit per channel, containing the processor, memory, and all channel interface electronics. Sensor, mode control, and servo interfaces are dedicated on a channel basis with data exchanged between computers via dedicated one-way, serial, optical, digital data buses that independently interconnect each computer to each other computer. Each computer has exclusive control over the engagement and shutdown of its own servos.

Key features of the computer unit design are a high throughput (420 kops) central processor; a computer I/O system in which the input and output processes are performed autonomously with no processor intervention required; and a solid-state memory partitioned into functionally independent program memory (ROM) and variable/scratch-pad memory (RAM). Input and output devices are embedded within the scratch-pad memory addressing structure with their own scratch-pad area and are therefore directly addressable by the processor.

Special hardware fault monitors, in addition to watchdog and servo monitors, include an arithmetic fault detector, RAM memory parity detector, digital I/O validity checks, and I/O loop testing provisions.

Real-time synchronization between computers is facilitated through the cross-channel exchange of synchronization discretes independently generated within each computer.

8.1.3 ARCS DESIGN ANALYSIS

The fault tolerance and cost/benefits of an ARCS were compared with a contemporary technology system to establish the advantages of the ARCS technology. Contemporary technology (i.e., fail-operational/fail-passive capability in a triplex configuration), used for example in the 747 analog Category III autoland system, was represented by the GE MCP 703 WWCS. The system concept was analyzed with respect to its fault-tolerant performance with two major objectives in mind: (1) to verify, through fault analysis, that the system concept, in fact, had the required fault-tolerant qualities from a functional point of view and (2) to assess, through a reliability analysis, the merit of those fault-tolerant qualities from a probabilistic point of view. The reliability analysis task had two main purposes: (1) to review the available reliability analysis methods and tools and to identify those suitable for use in the ARCS study and (2) to assess the reliability of the ARCS design and configuration alternatives within the scope of the defined operational applications. An assessment of the cost effectiveness of applying ARCS technology was carried out in two parts: an analysis of airline cost-of-ownership for an ARCS maintained in a Cat III operational status and an analysis of the cost effect of providing an integrated

system test function that could be used to give line replaceable unit—sensor, computer, and servo—failure identification for line maintenance.

Simulation was used as a tool in the fault analysis as we began to evaluate the reconfiguration processes. The simulation demonstrated that the ARCS fault-tolerant design was a valid concept, even though the sensor signal selection/failure detection algorithms to deal with the duplex-to-simplex operation were not fully developed.

A new computer-aided redundant system reliability analysis tool (CARSRA) was developed. CARSRA advances the current technology in reliability assessment by modeling effects such as failure “coverage,” transient faults, and intermodule dependencies, and possesses the capability to compute system functional readiness (availability) and failure probability. “Coverage” is the probability that the system continues to operate given a failure.

CARSRA uses a unique modeling technique, which consists of partitioning the system into stages where a stage is defined as a set of identical redundant modules. The possible functional redundancy state of each stage is described by a Markov model, and failure dependencies between stages are specified through a special entry to the CARSRA program. This unique modeling technique makes it possible to describe nuisance failures and failure coverage effects in the Markov model for each stage, thereby avoiding the problem of having to deal with the exorbitant number of states that would result if the complete system were to be modeled by a single Markov model.

The emphasis of the transient fault analysis effort was directed toward the prediction of sensor nuisance failures since these have been a major problem in contemporary systems. The analysis indicated that only a few particular sensors are potentially troublesome and that, with careful design, nuisance failure indications could be controlled even for these sensors.

CARSRA was used to generate projected reliability data for an ARCS baseline and several alternate configurations. It was shown that, assuming present-day failure rates, a quadruplex ARCS-type system would meet the fly-by-wire reliability requirement, while a conventional fail-op/fail-op/fail-passive system would fall short. An ARCS-implemented control/stability augmentation function has a five times lower failure probability than does the same function using the WWCS.

The probability of a system failure during a Cat III autoland, assuming that all system modules are functional at the alert height, was 10 times lower for the ARCS than for the WWCS. An advanced sensor system would result in an additional factor of 6 improvement with the ARCS. Both the ARCS and WWCS failure probability projections surpass the Cat III autoland requirements by a wide margin. However, the ARCS was shown to also meet these requirements with one failure existing at the alert height.

Requiring that all system modules be functional at the Cat III alert height imposes a significant penalty relative to the functional availability of autoland. By permitting an autoland to be initiated with one module failed (sensor, computer, or servo), ARCS achieves a 0.93 availability, assuming a 100-hour maintenance interval. In comparison with the conventional system (TMR) availability of 0.73, this means a factor of 4 reduction in diversions due to Cat III autoland unavailability for the ARCS in comparison with the WWCS.

A voting node placement trade study showed that a significant improvement (a factor of 10) in system functional reliability is achieved by providing a sensor signal voting node to a "brickwall" servo output voted configuration. A negligible improvement is obtained, however, when an additional voting node is introduced between the computers and servo drive electronics.

A new approach was developed for measuring "coverage" (likelihood of survival given a fault) that is based on a failure analysis of a randomly selected set of failure modes extracted from the entire failure mode population. This approach, shown to be feasible by an analysis/laboratory experiment, may provide a cost-effective method for demonstrating a fault-tolerant system's potential functional success probability.

Three primary factors were considered in conducting the cost/benefit analyses: acquisition cost, maintenance cost, and costs associated with schedule interruptions caused by not having an operational Cat II/Cat III system. The cost data for the former analysis was compiled by comparing the ARCS with the WWCS, which represented a contemporary triple-modular-redundant (fail-op/fail-passive) system design. The data base for the latter study was the ARCS hardware without the maintenance enhancement elements, maintained in a conventional manner.

Using the defined airline model (150 aircraft) and a specific route structure typical of United's 727 fleet, the ARCS-equipped airline can be expected to incur approximately 69 fewer diversions per year because of weather conditions requiring low-visibility landing capability. Based on an average cost per diversion this can save approximately \$145 000 annually. Low-visibility operation is not standard procedure with all airlines today. Category III autoland in particular is limited to a few airlines and airports worldwide. Low-visibility capability is not a dispatch requirement but an airline option.

Including the system test maintenance feature in the ARCS can result in a potential further annual saving of approximately \$150 000 for the same airline model. This cost saving is brought about largely by improving the effectiveness (success probability) of the maintenance action, thereby reducing the number of unwarranted equipment removals experienced with today's systems.

The cumulative effect of system acquisition cost and avoided diversion expense, for the defined airline model, is an annual cost saving of approximately \$495 000, more than \$3000 per aircraft. The inclusion of system test yields a further saving of approximately \$1000 per aircraft per year.

Other cost savings attributable to the ARCS can be expected as we look deeper into the cost-of-ownership picture—reduced spares, maintenance procedural improvements, dispatch availability, etc.

8.2 CONCLUSIONS

The ARCS program assessed, in a broad sense, the relevance of fault-tolerant computer technology to commercial jet transport avionics applications. The overall conclusion of this assessment is that the economic impact of the ARCS technology is evolutionary rather

than revolutionary. The full benefit of fault-tolerant computer technology will be reaped when the functions provided by the avionic systems become dispatch critical or otherwise mandated for use on every flight. The greater functional survivability and availability result from reconfiguration to simplex. For some users, simplex may not be an acceptable mode of operation.

A significant accomplishment of the ARCS analysis task was the development of a reliability analysis capability, including a computerized tool, that considerably improves the ability to analyze redundant system architectures. The CARSRA program proved to be a powerful tool essential for ARCS-type reliability assessments. The functional availability analysis was an eye-opener in the sense that it showed that a TMR-type system will not achieve postulated autoland functional availability goals.

All aspects of duplex-to-simplex reconfiguration were not exhaustively covered by the ARCS study; work remains in the areas of analytical redundancy (fault monitoring of simplex signals) and techniques for predicting and evaluating "coverage"; i.e., the likelihood of function survival given a fault in the duplex state. Assessment of "coverage" is a central problem in designing fault-tolerant systems. Credible reliability estimations hinge on the level of confidence at which "coverage" values can be estimated for duplex to simplex operations. Application of an ARCS-developed method, based on random-fault insertion, to a gate level simulation/emulation of a candidate computer is seen as the next step in fault-tolerant technology consolidation.

The ARCS data supports application of fault-tolerant computer technology to increase the effectiveness of flight-critical avionics at lower cost to the operator. Fault-tolerant computing will be a significant element in achieving fly-by-wire and active controls technology, as well as general use of Cat III capability, at acceptable cost.

Boeing Commercial Airplane Company
P. O. Box 3707
Seattle, Washington 98124
August 4, 1976

APPENDIX A

ARCS APPLICATION MODEL CONTROL LAWS

This appendix describes control laws representative for an ARCS application. Figure A1 is a reference overview to the individual pitch, roll, yaw, flutter and maneuver load control laws for the autoland, go-around and CCV/FBW functions which are presented in Figures A2 through A10.

The exact complement of functions will depend on the specific vehicle under consideration. The ARCS Near Term application model was assumed to include the functions described by Figures A2 through A5, plus A8; the Intermediate Term the functions in Figure A2 through A8; and the Far Term model all of the described functions.

Sensor and mode control interface requirements, and servo and display interface requirements for the near, intermediate and far term application models are shown in Figures A11 through A16.

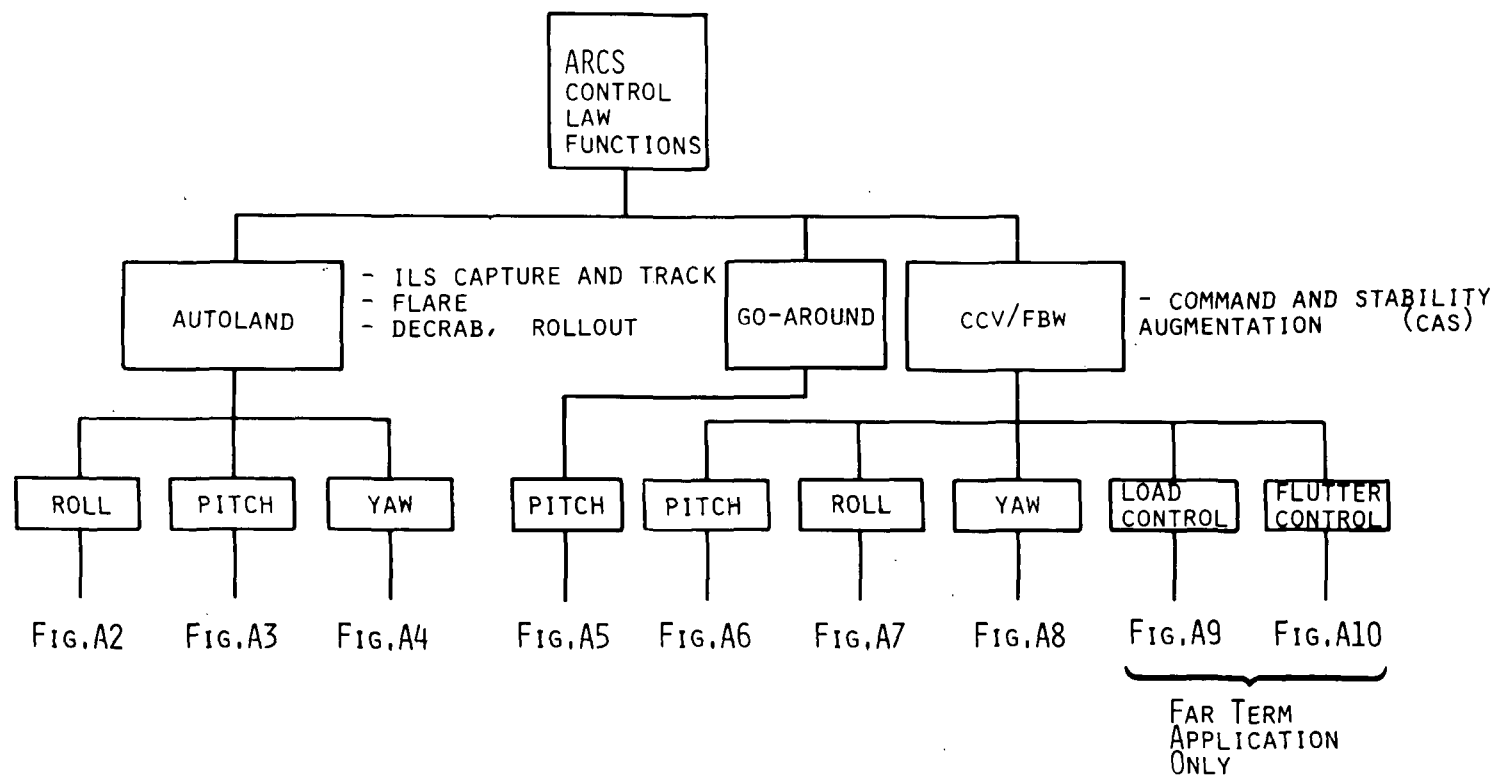


FIGURE A1 APPLICATION CONTROL LAW OVERVIEW

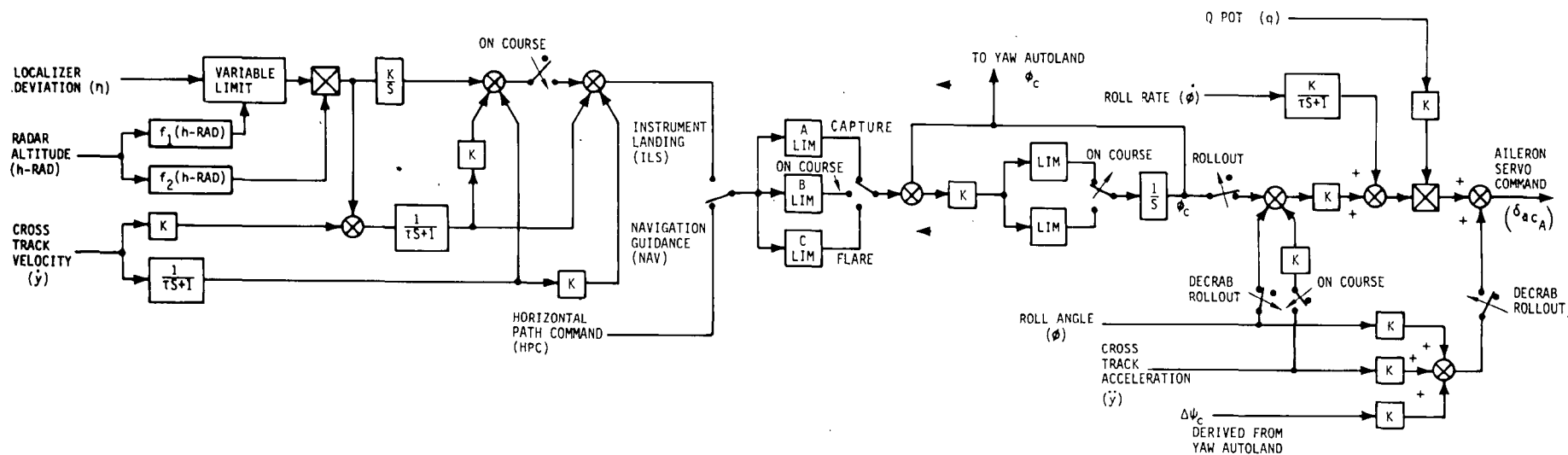
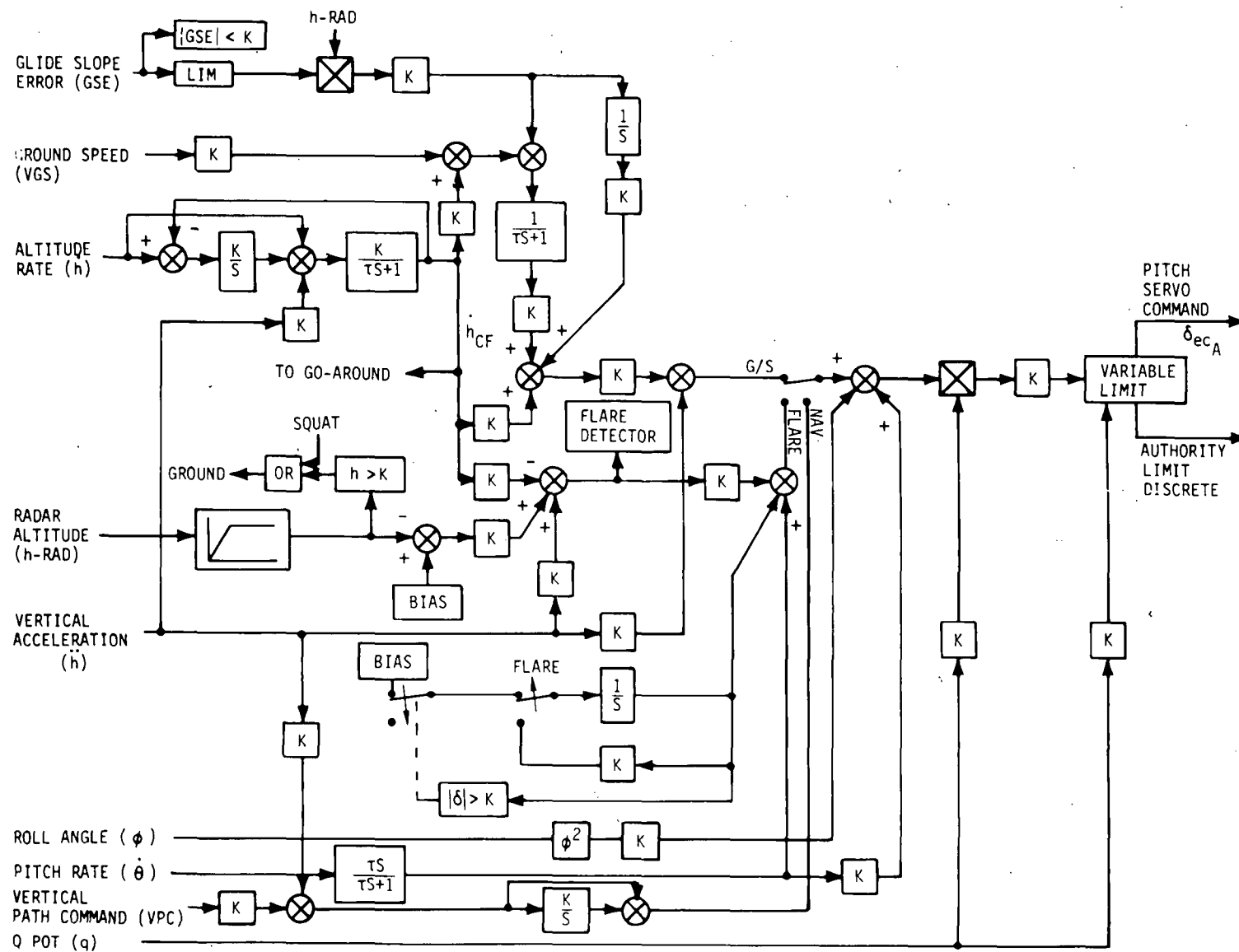


FIGURE A2 ROLL AUTOLAND CONTROL LAW



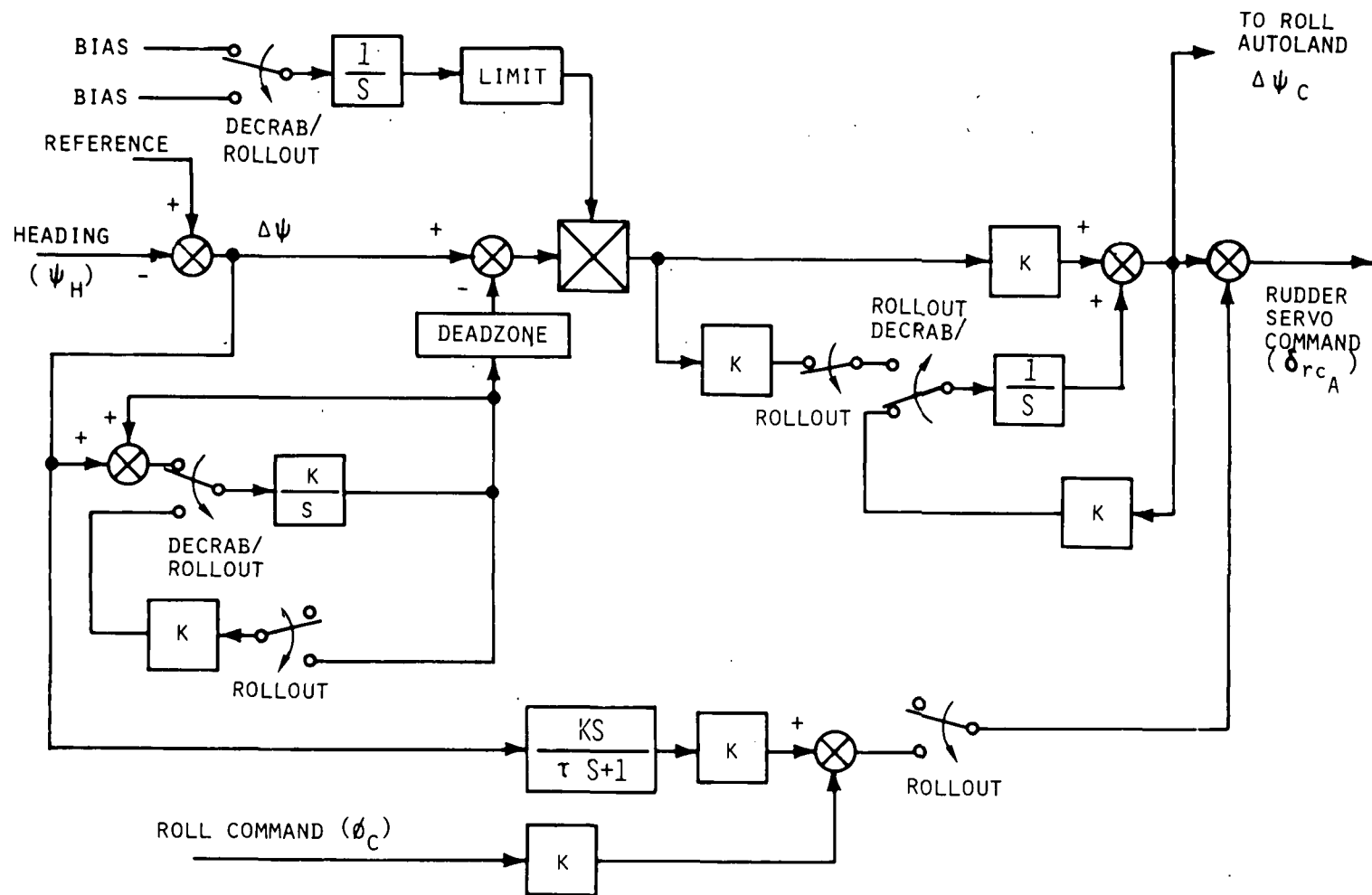


FIGURE A4 Yaw AUTOLAND CONTROL LAW

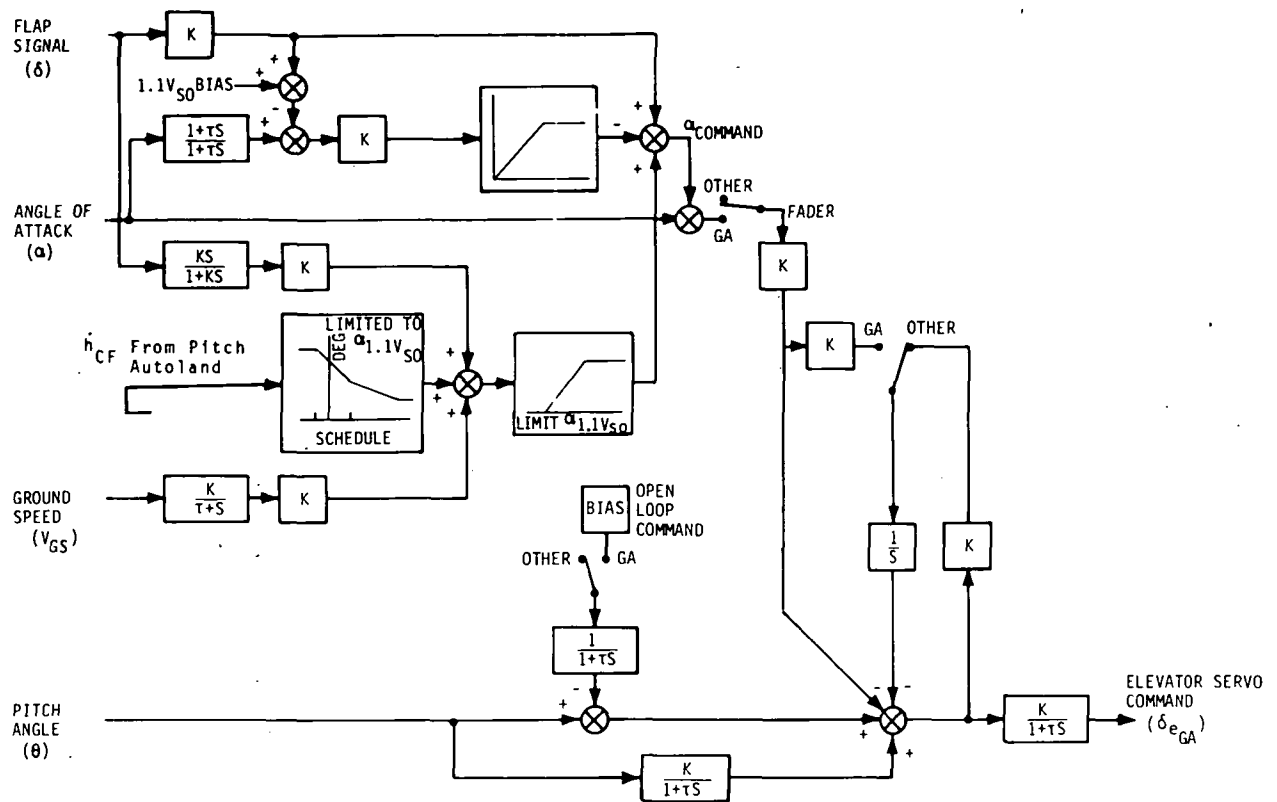


FIGURE A5 Go-Around Control Law

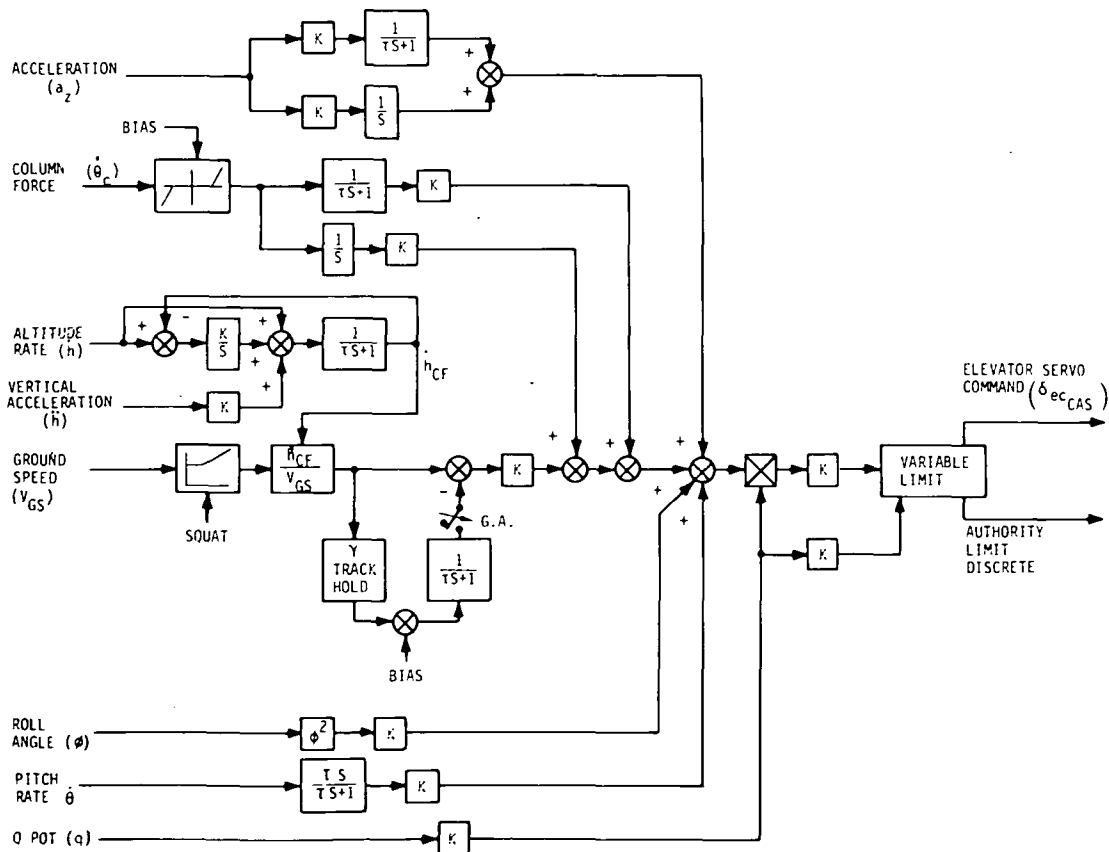


FIGURE A6 Pitch Command Augmentation Control Law (With Y Hold)

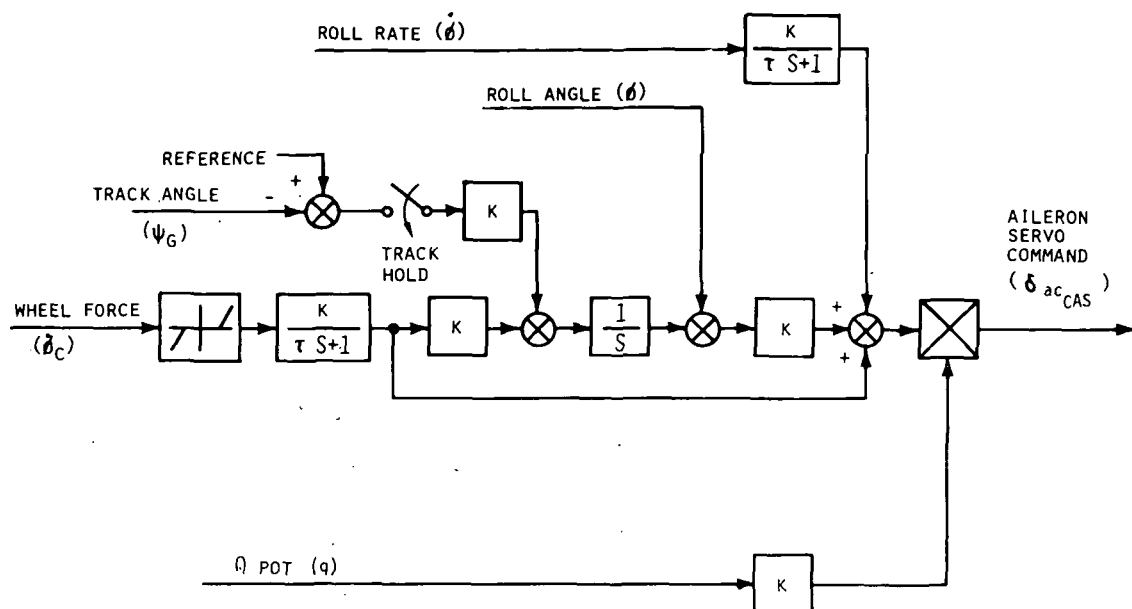


FIGURE A7 ROLL COMMAND AUGMENTATION CONTROL LAW

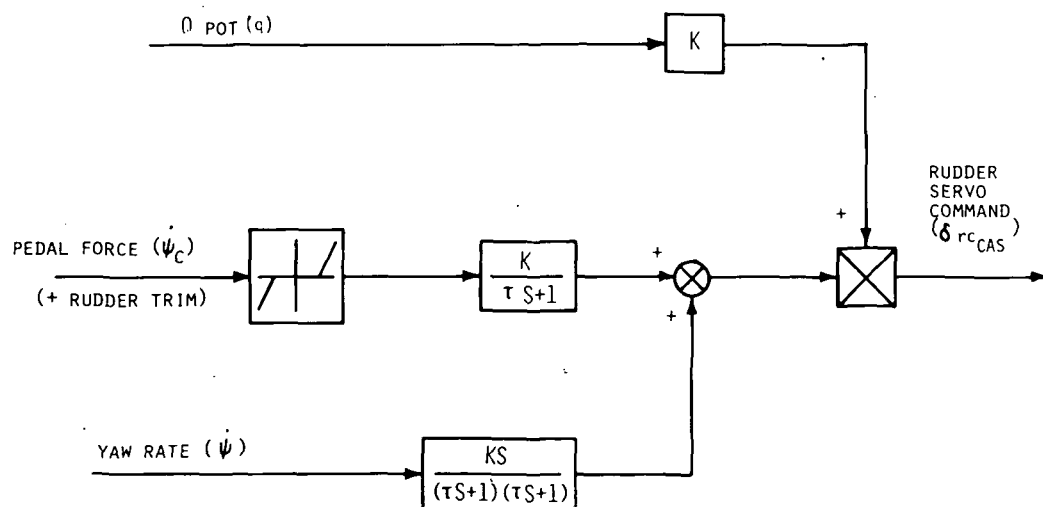


FIGURE A8 YAW COMMAND AUGMENTATION CONTROL LAW

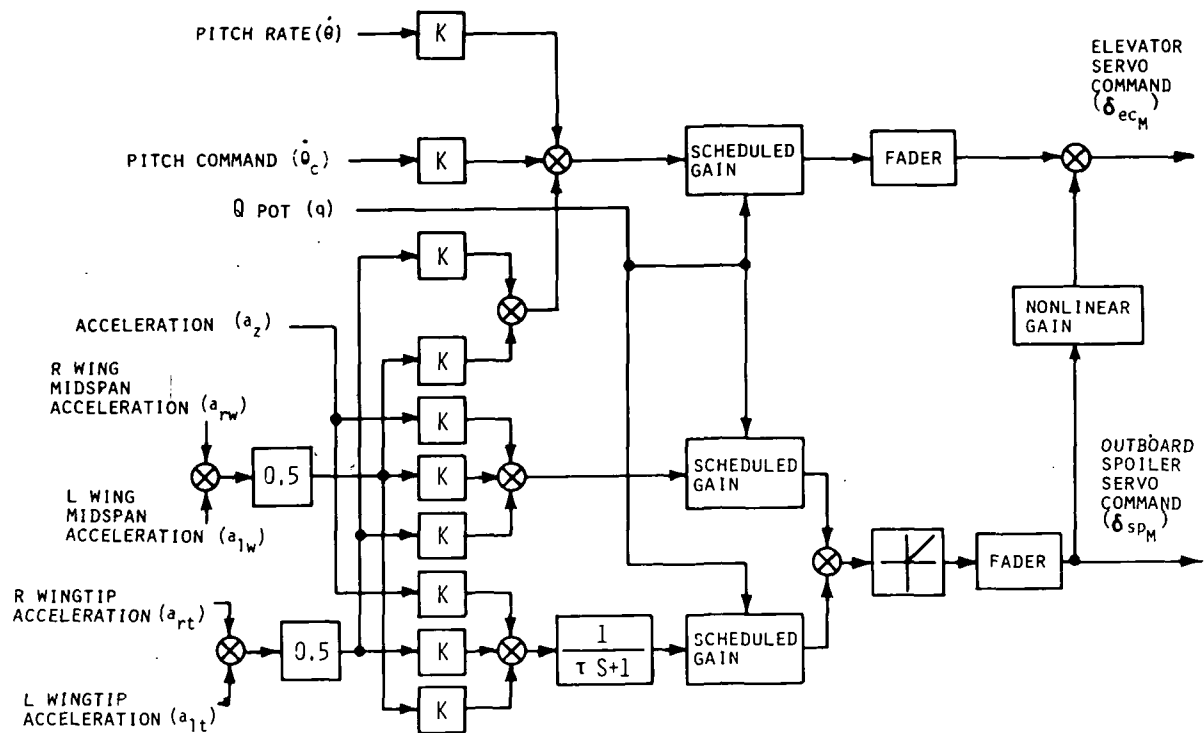


FIGURE A9 MANEUVER AND GUST LOAD ALLEVIATION CONTROL LAW

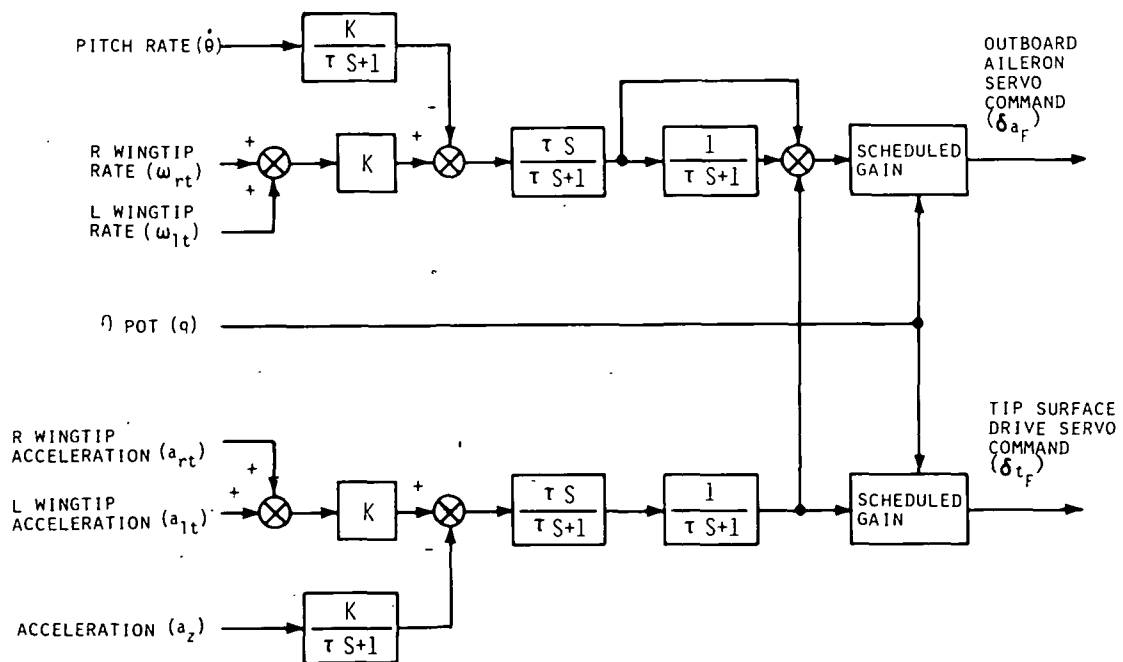


FIGURE A10 FLUTTER SUPPRESSION CONTROL LAW

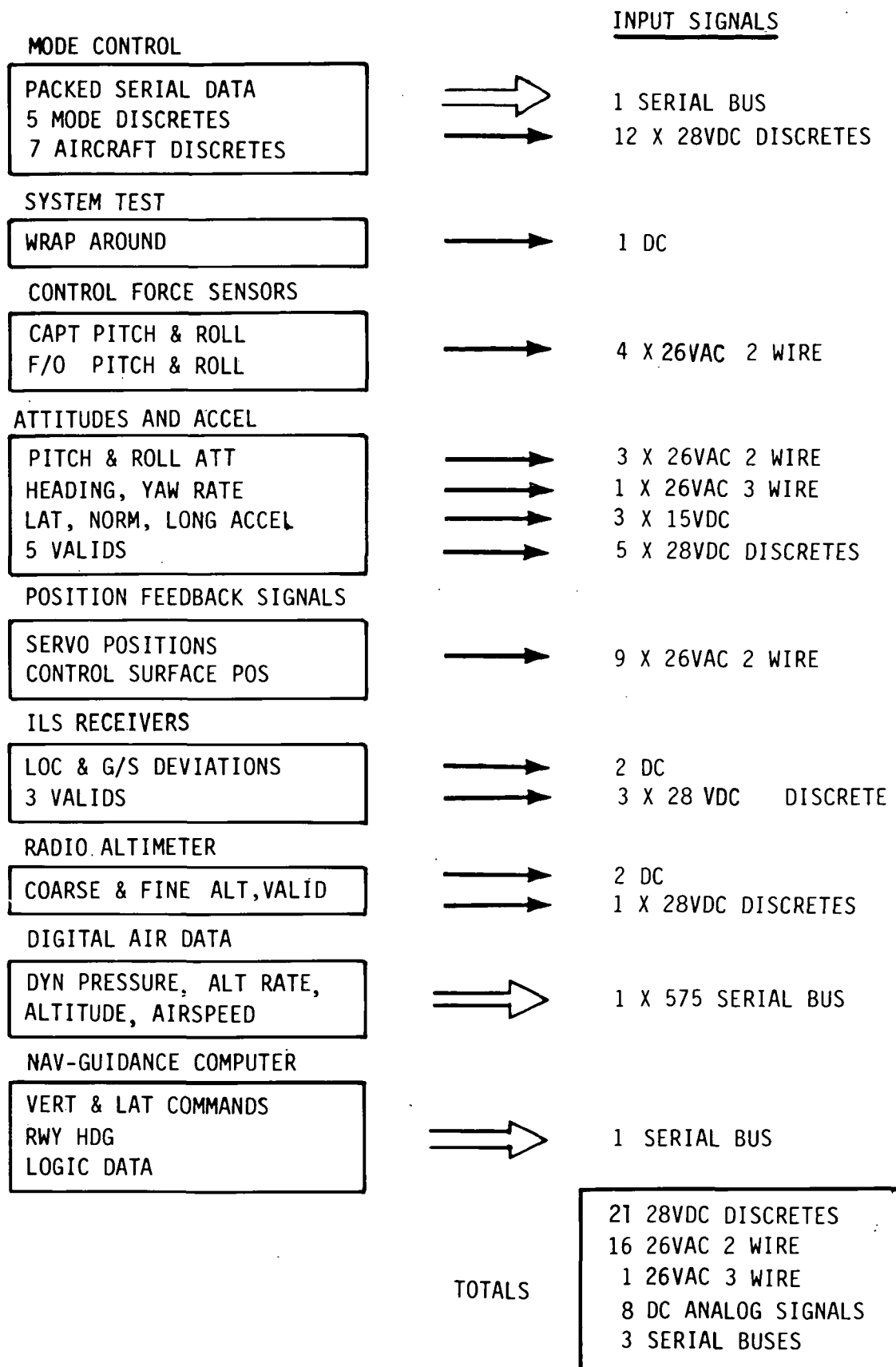
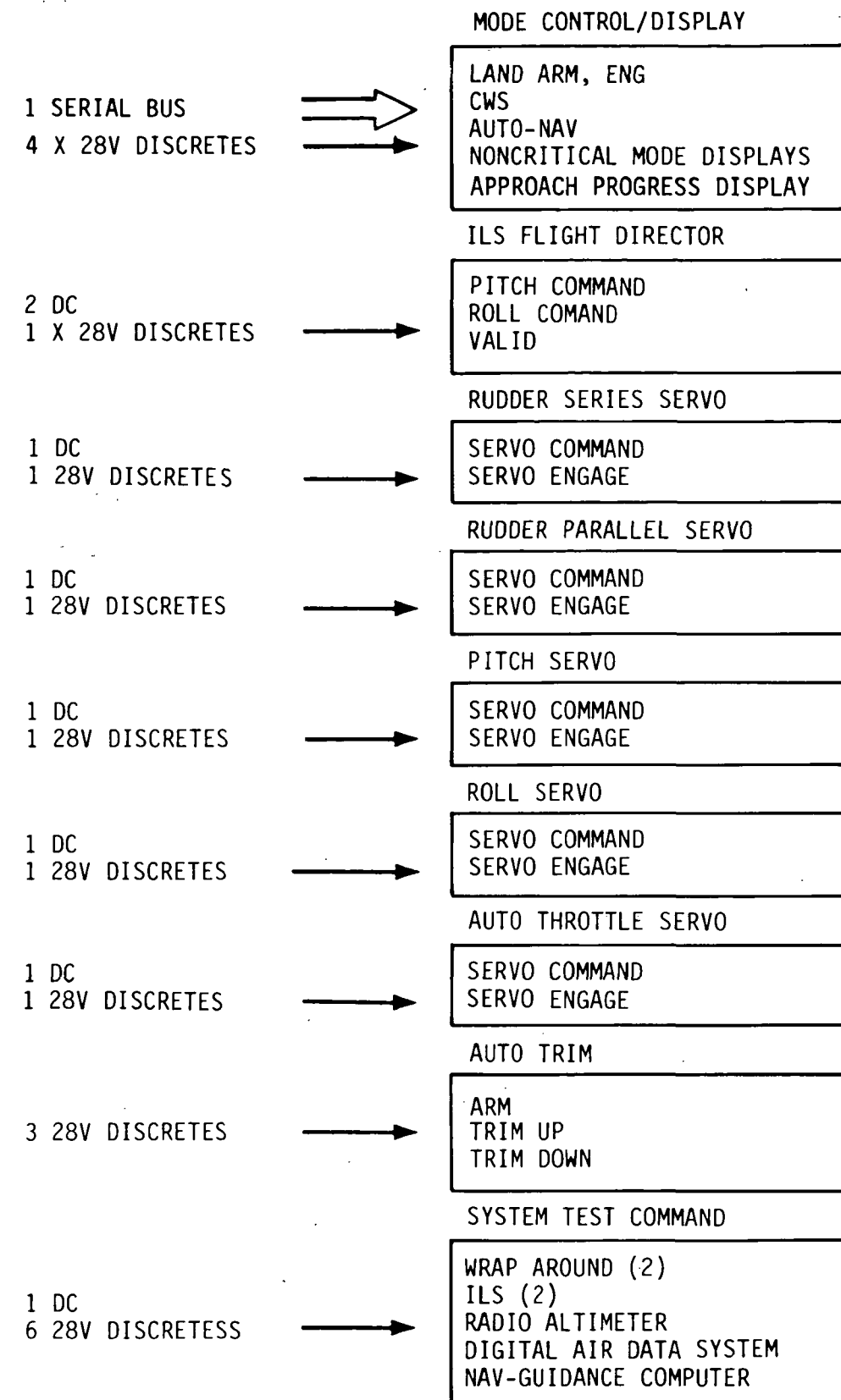


FIGURE A-11 SENSOR AND MODE CONTROL BLOCK —
NEAR TERM APPLICATION MODEL



SUMMARY

19 28V DISCRETES
8 DC
1 SERIAL BUS

FIGURE A-12 SERVO AND DISPLAY BLOCK —
NEAR TERM APPLICATION MODEL

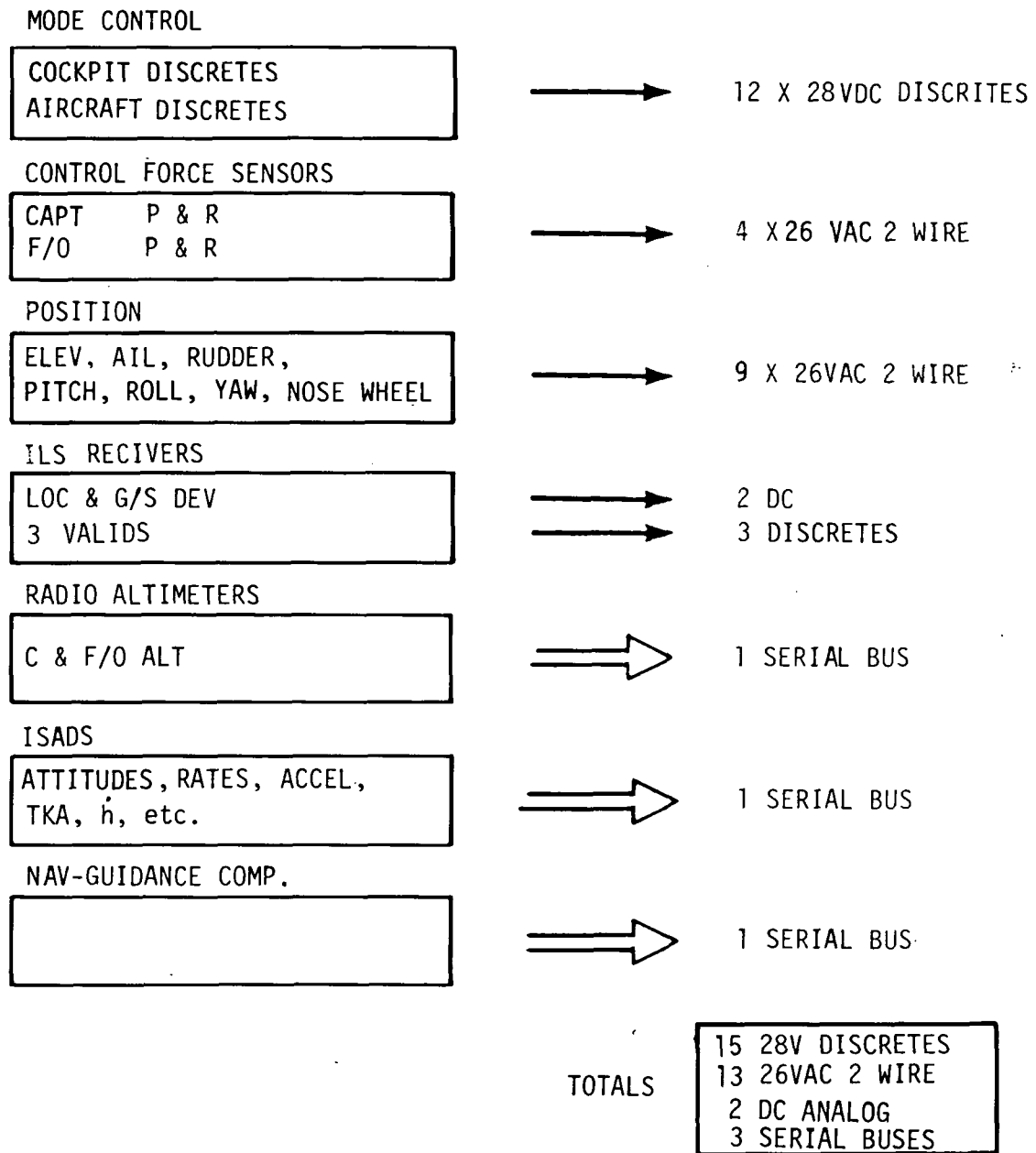


FIGURE A-13 SENSOR AND MODE CONTROL BLOCK —
INTERMEDIATE TERM APPLICATION MODEL

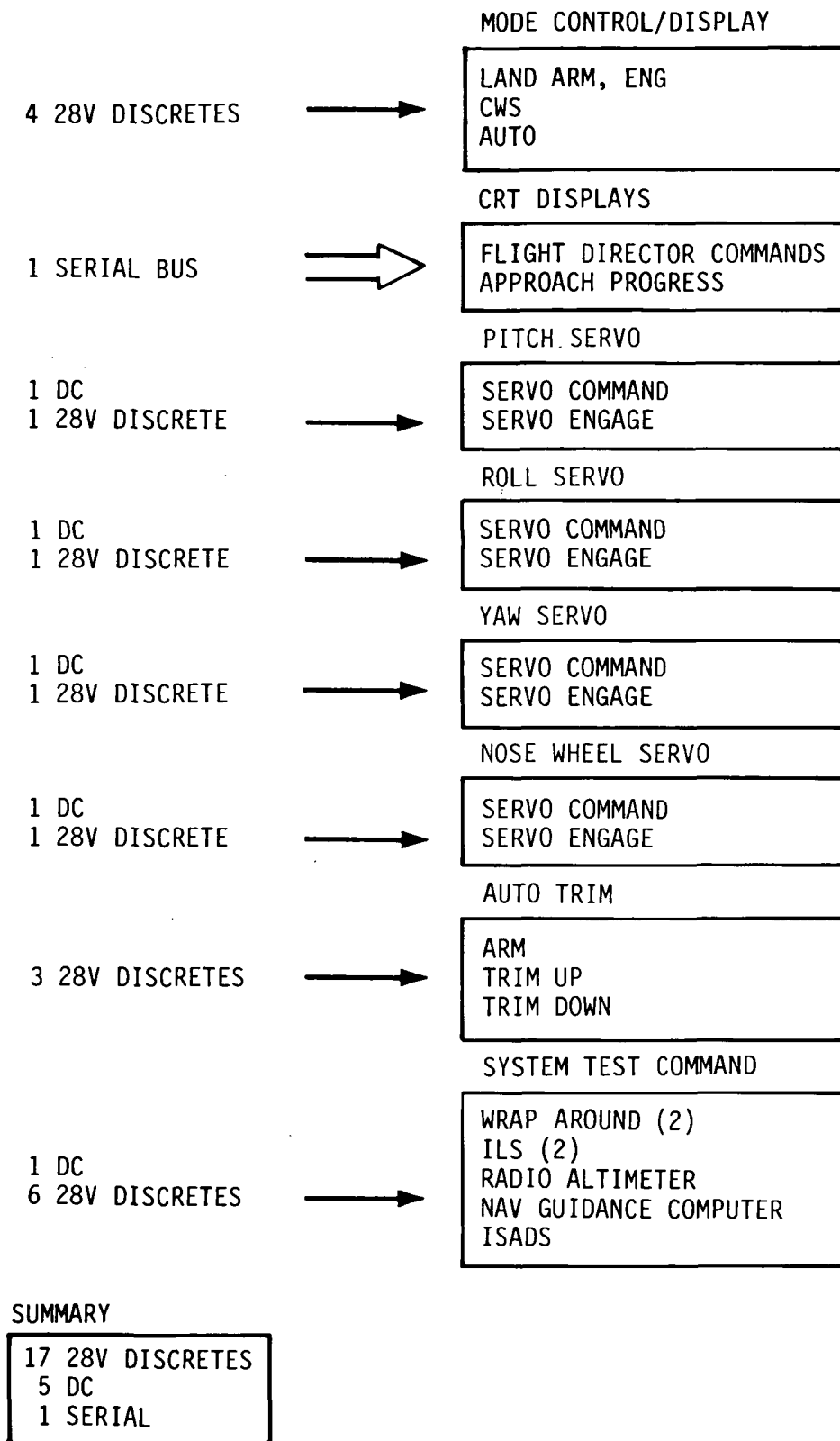


FIGURE A-14 SERVO AND DISPLAY BLOCK
INTERMEDIATE TERM APPLICATION MODEL

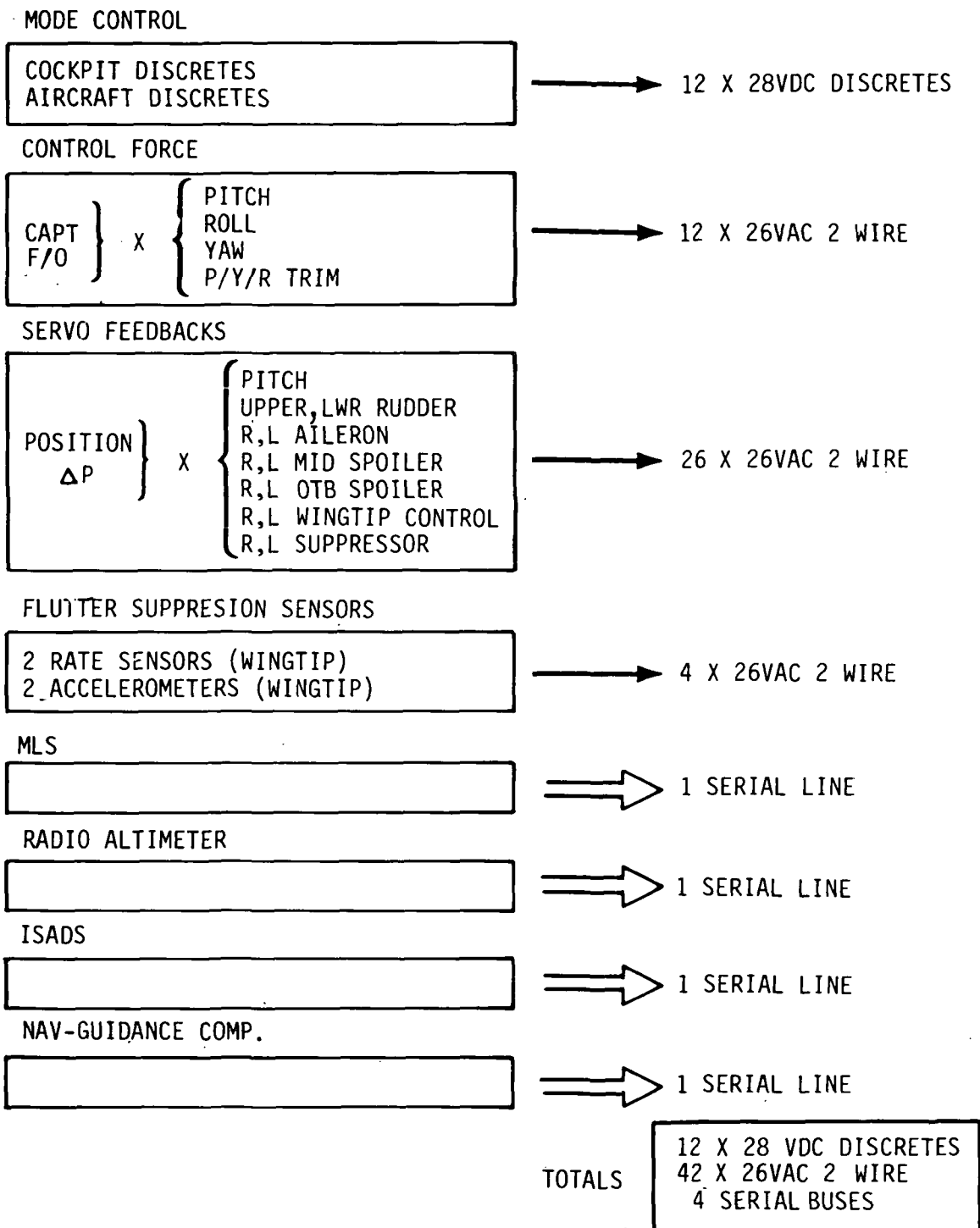


FIGURE A-15 SENSOR AND MODE CONTROL BLOCK -
FAR TERM APPLICATION MODEL

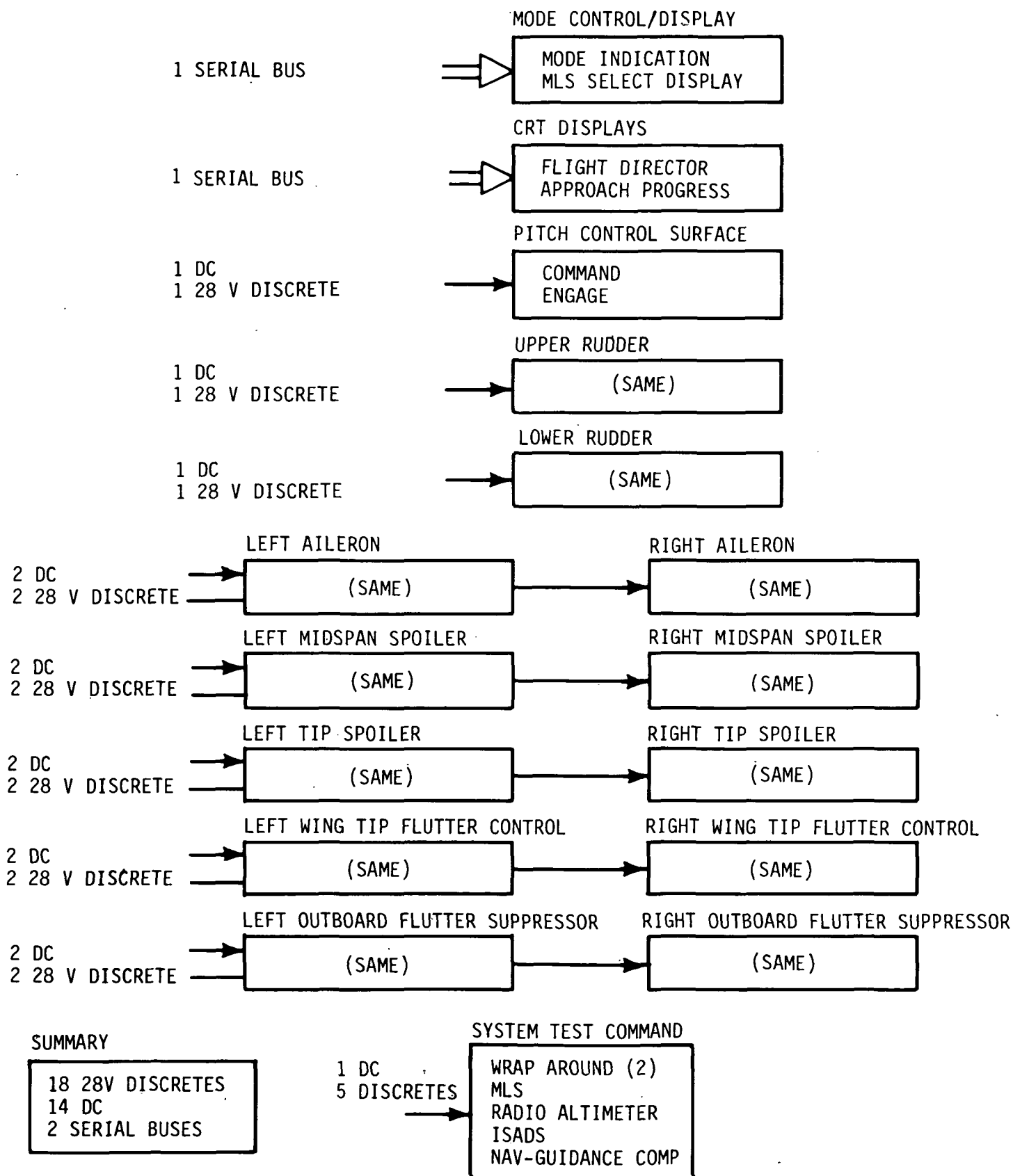


FIGURE A-16 SERVO AND DISPLAY BLOCK —
FAR TERM APPLICATION BLOCK

APPENDIX B

SOFTWARE DESCRIPTION

B.1 ARCS SOFTWARE DESCRIPTION

This appendix describes the ARCS baseline software whose functional tree is shown in Figures B1-A, B1-B, B1-C, and B1-D. The software structure tree, shown in Figures B2-A, B2-B, B2-C, and B2-D, is derived from the Functional Tree and gives a short mnemonic name to each function.

The executive function of the ARCS software is contained within the module-to-module flow of control shown by the state transition diagrams of Figures B3-A and B3-B. Process iteration is governed by the minor frame schedule shown by the table in Figure B4, where a major frame is subdivided into two minor frames of 20 ms for the near term and four minor frames of 10 ms for the far term models. This major frame subdivision facilitates scheduling processes requiring iterations of 50 times a second (command augmentation functions) and 20 times a second (autoland path guidance control functions).

The software module descriptions that follow are presented using the format introduced in the software design methodology. Subsection B.1.1 describes those software modules associated with the high level functions of the ARCS functional tree (reference Figure B1-A), i. e., real time, ground test, interrupt

processing synchronous (foreground), and asynchronous (background) executive relationships. Subsection B.1.2 describes the heart of the ARCS real time foreground functions, the redundancy management processes covered by Figure B1-D.

Subsection B.1.3 contains the software module descriptions of the application control laws and associated mode control. Subsection B.1.4 contains the software module descriptions of the background task. Subsection B.1.5 contains the software module description of the ground test.

B.1.1 ARCS High Level Software Modules

To facilitate the use and maintenance of software developed according to the ARCS software design format each new module description should start at the top of a page. With two exceptions, this procedure has been followed throughout the following module descriptions.

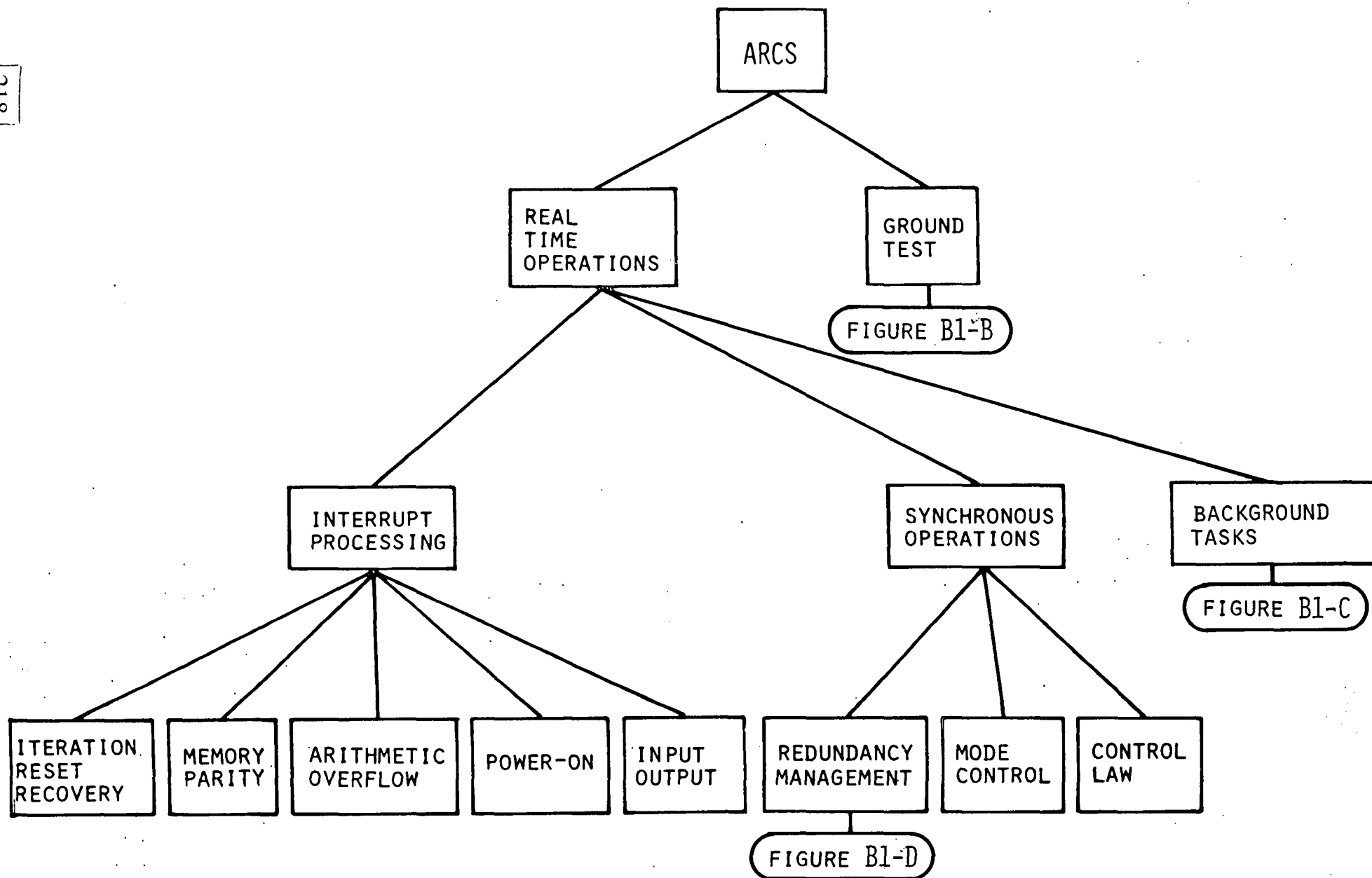


FIGURE B1-A ARCS FUNCTIONAL TREE

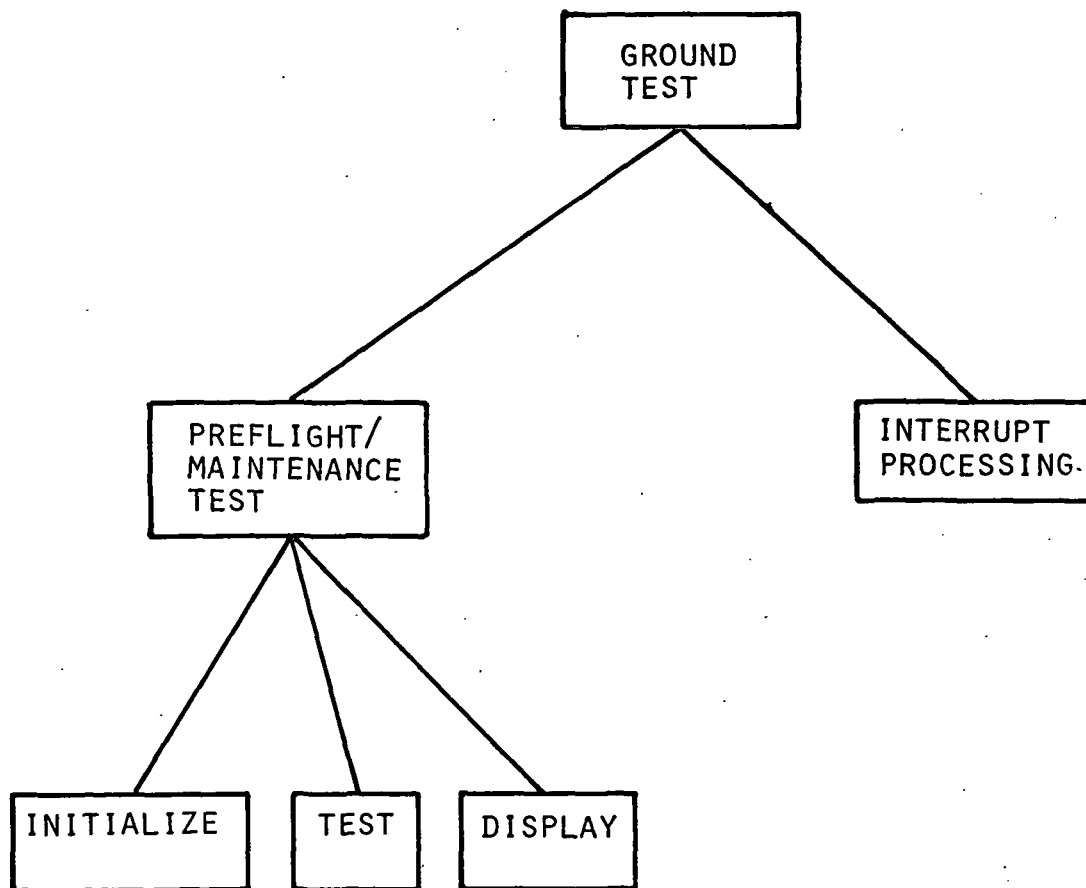


FIGURE B1-B GROUND TEST FUNCTIONAL REQUIREMENTS

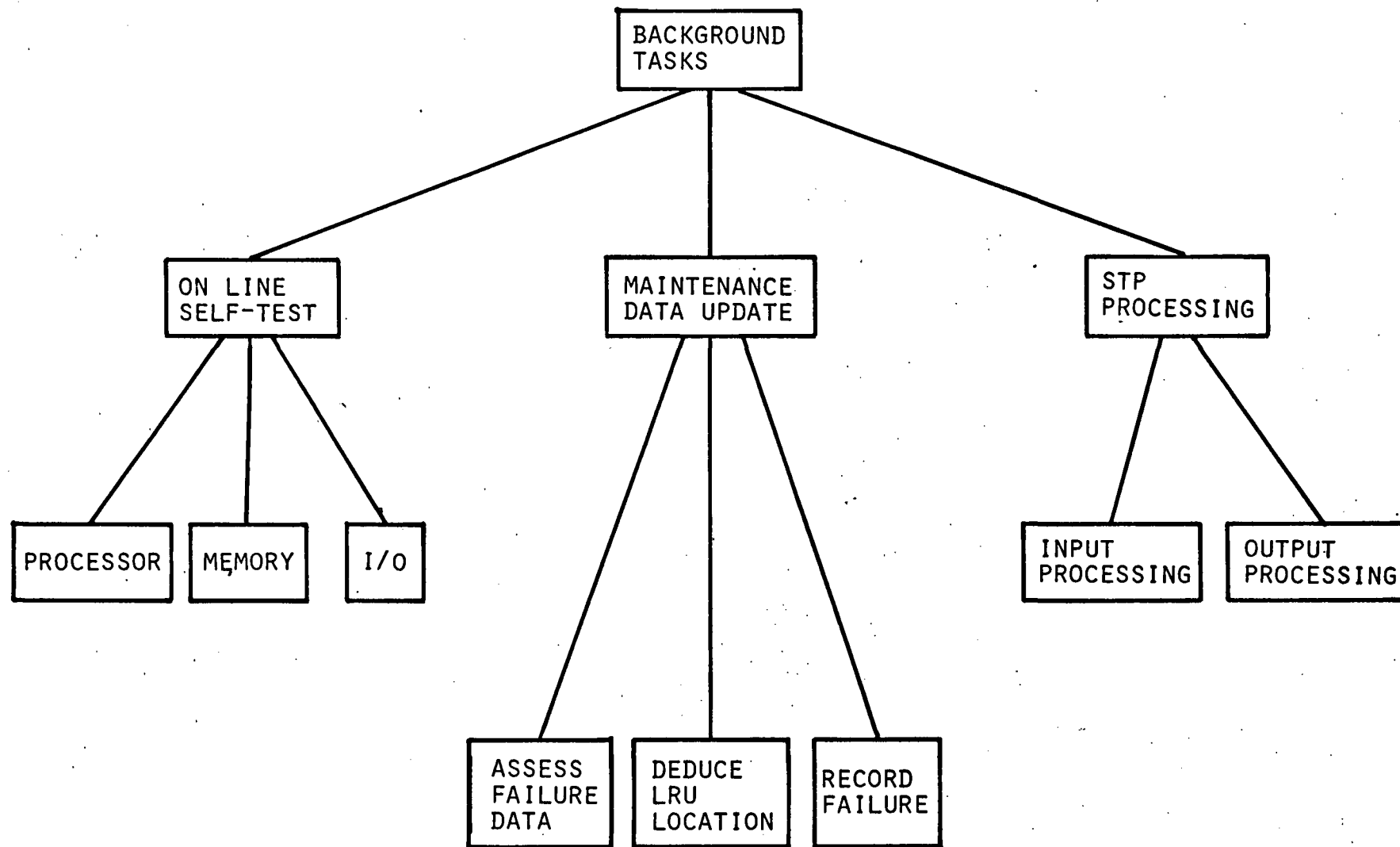


FIGURE B1-C BACKGROUND TASKS REQUIREMENTS TREE

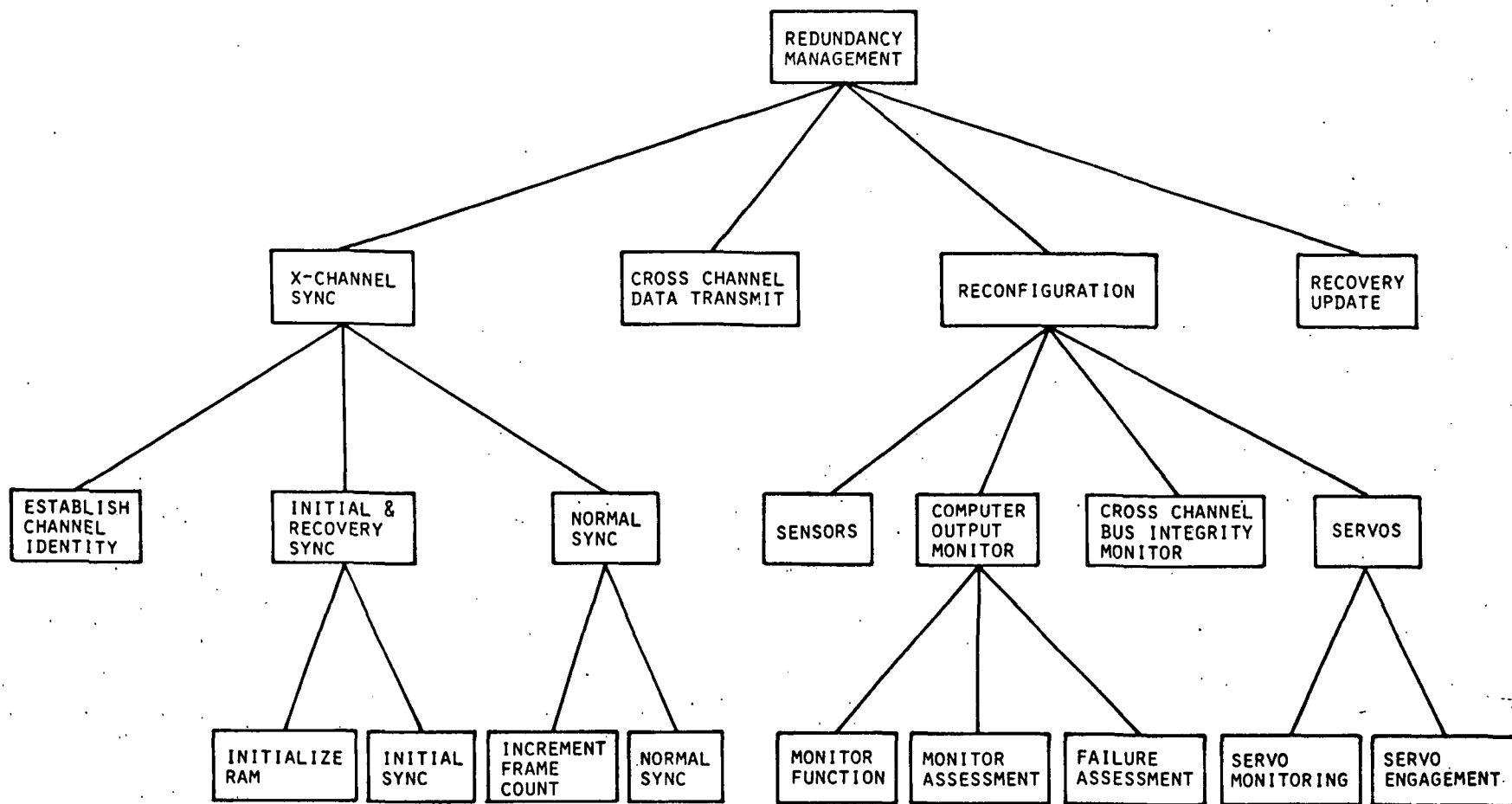


FIGURE B1-D ARCS REDUNDANCY MANAGEMENT FUNCTIONAL TREE

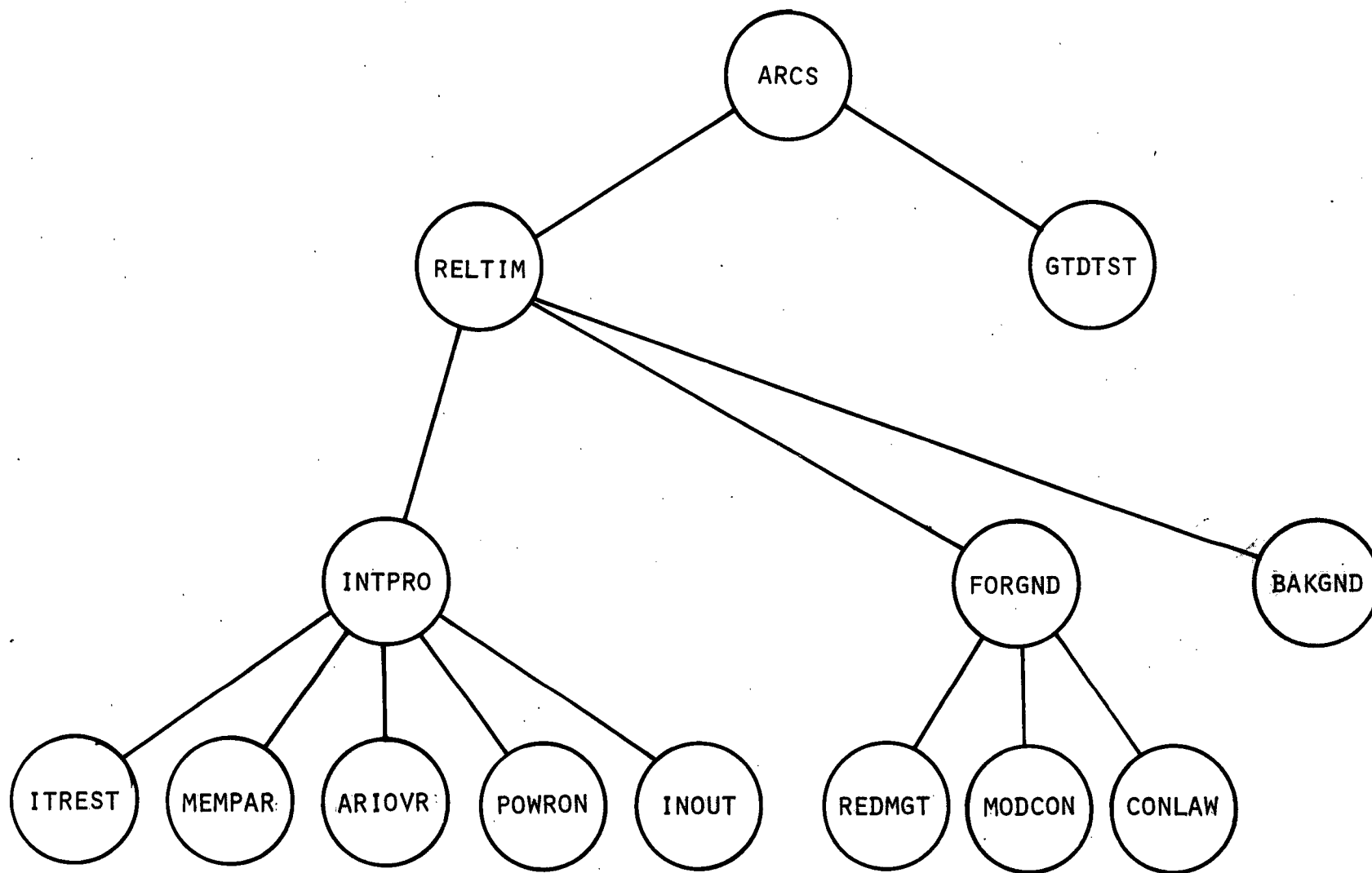


FIGURE B2-A ARCS SOFTWARE STRUCTURE

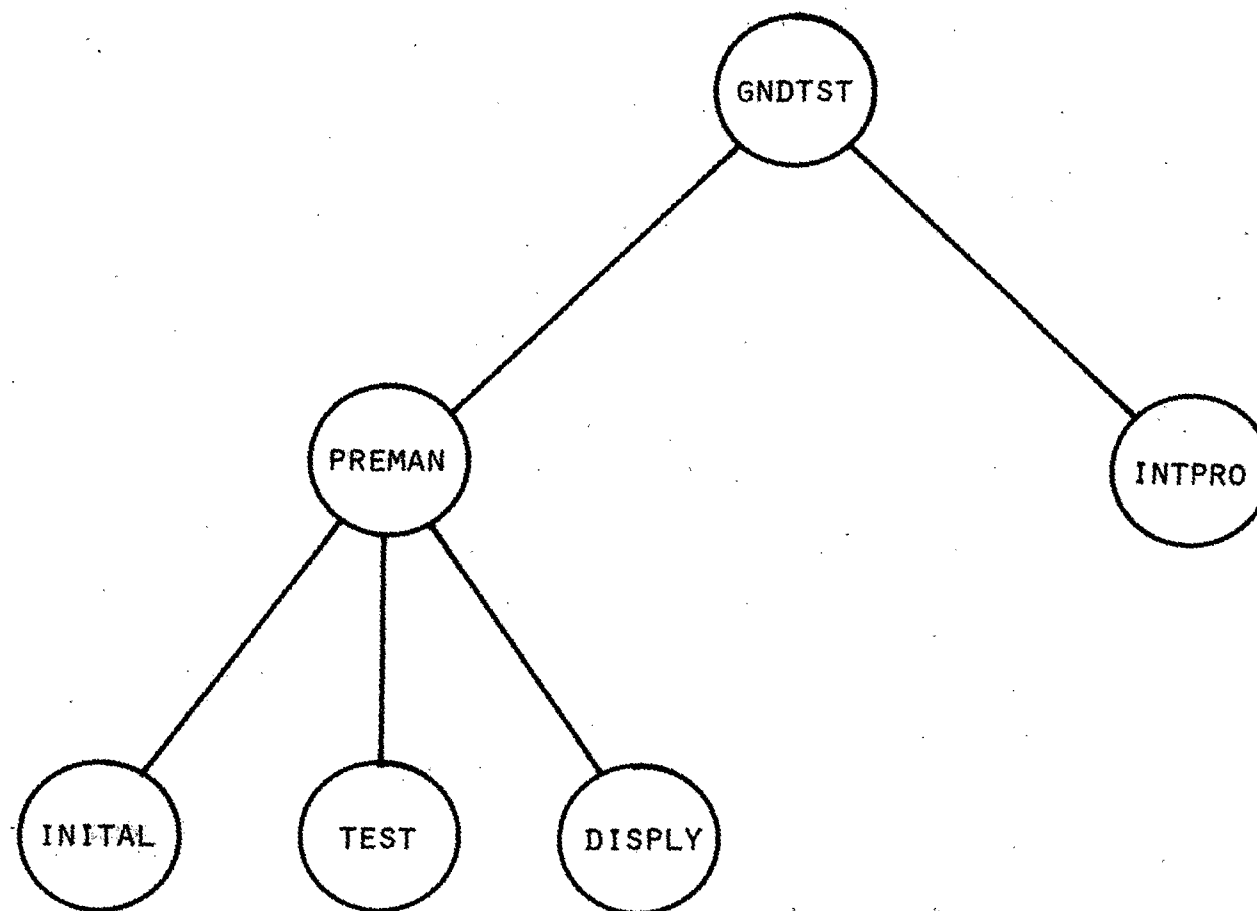


FIGURE B2-B ARCS SOFTWARE STRUCTURE

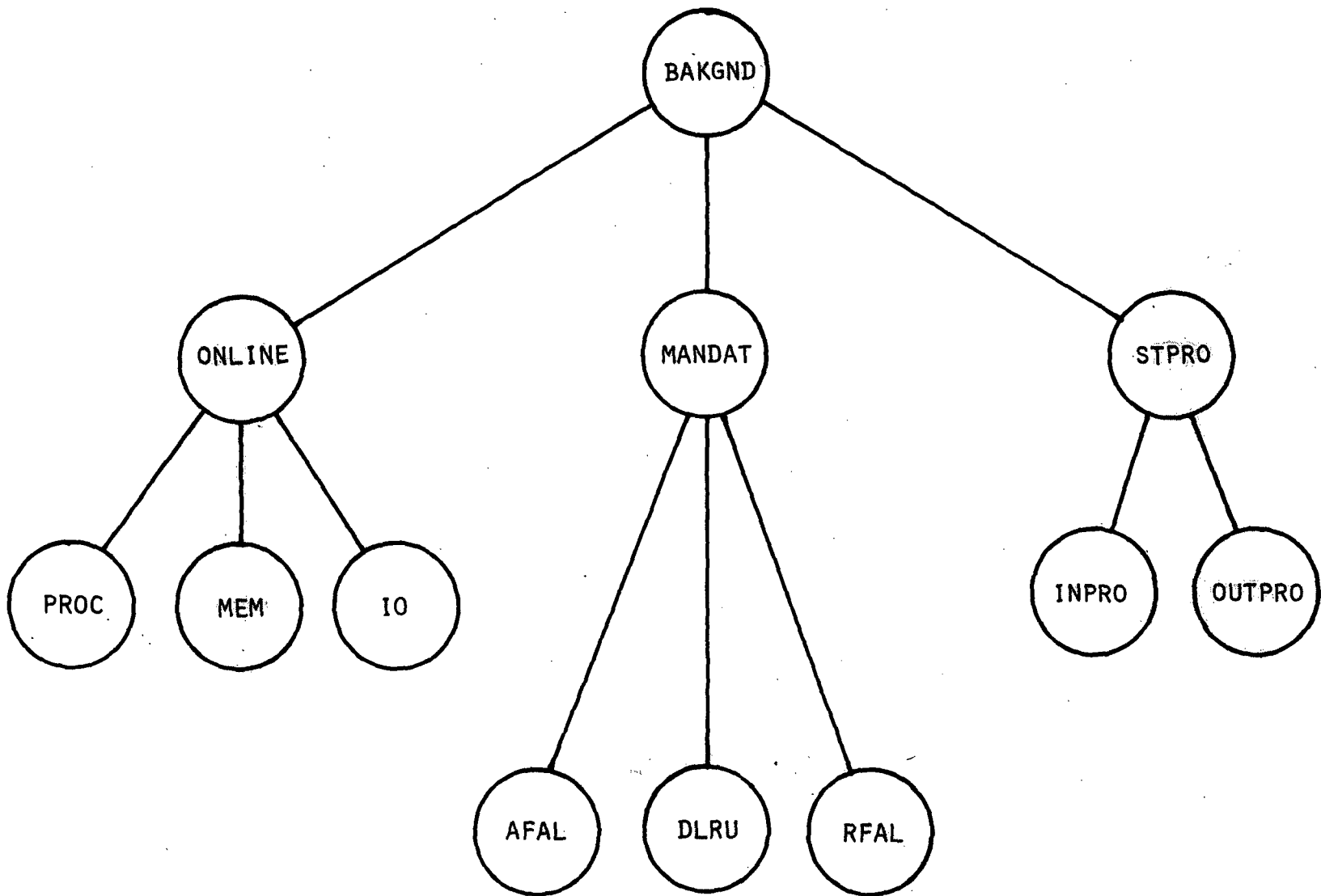


FIGURE B2-C ARCS SOFTWARE STRUCTURE

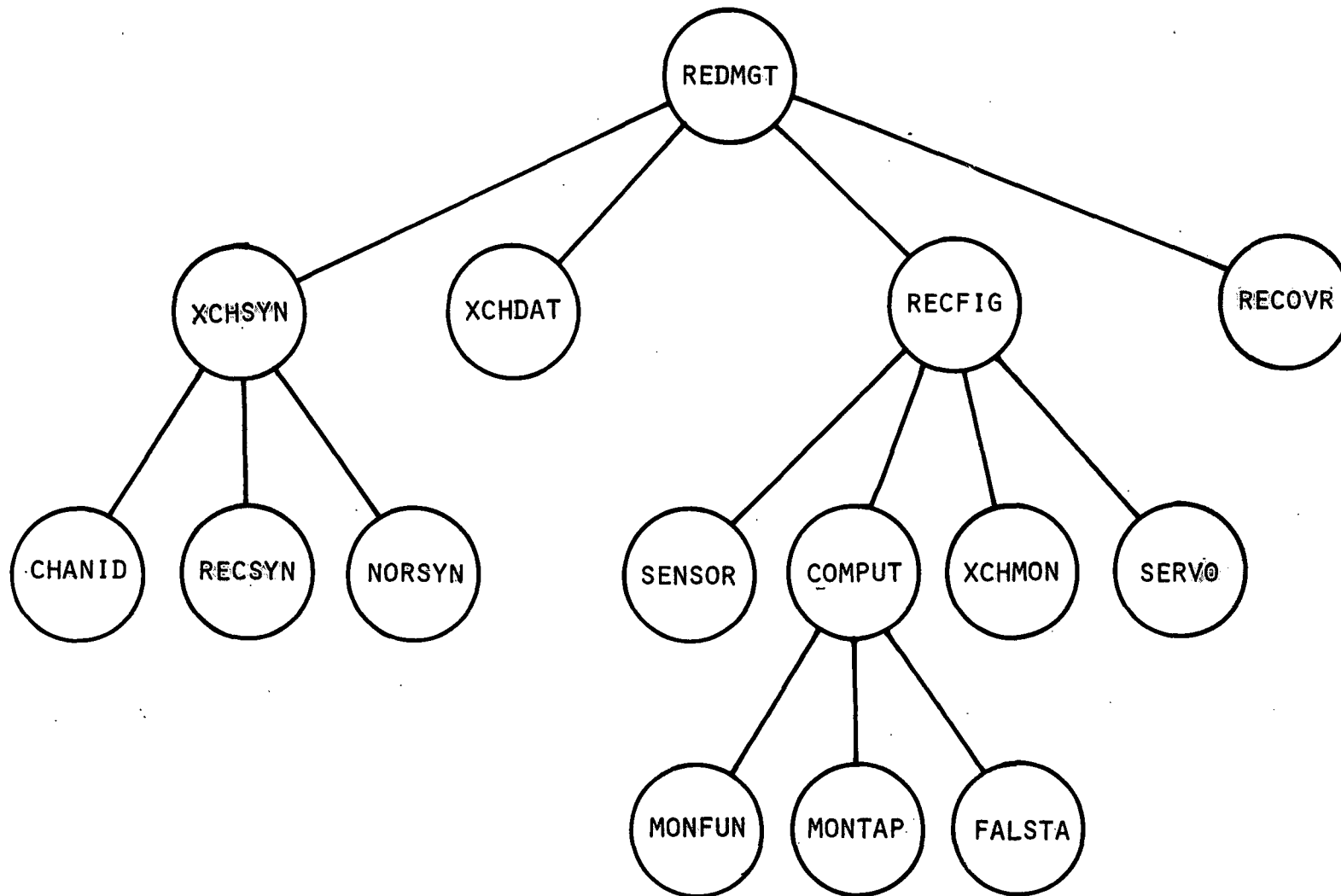


FIGURE B2-D ARCS SOFTWARE STRUCTURE TREE

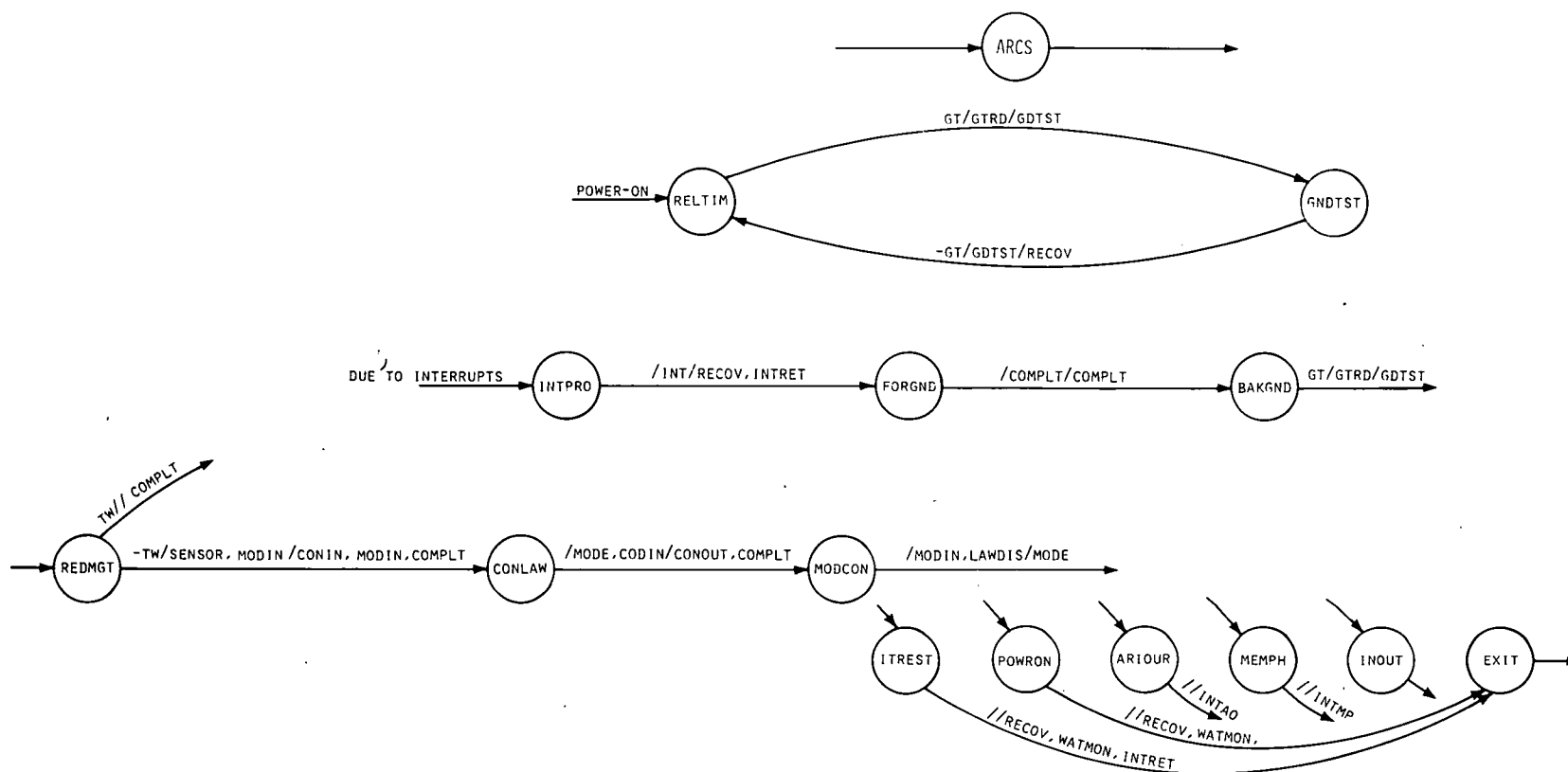
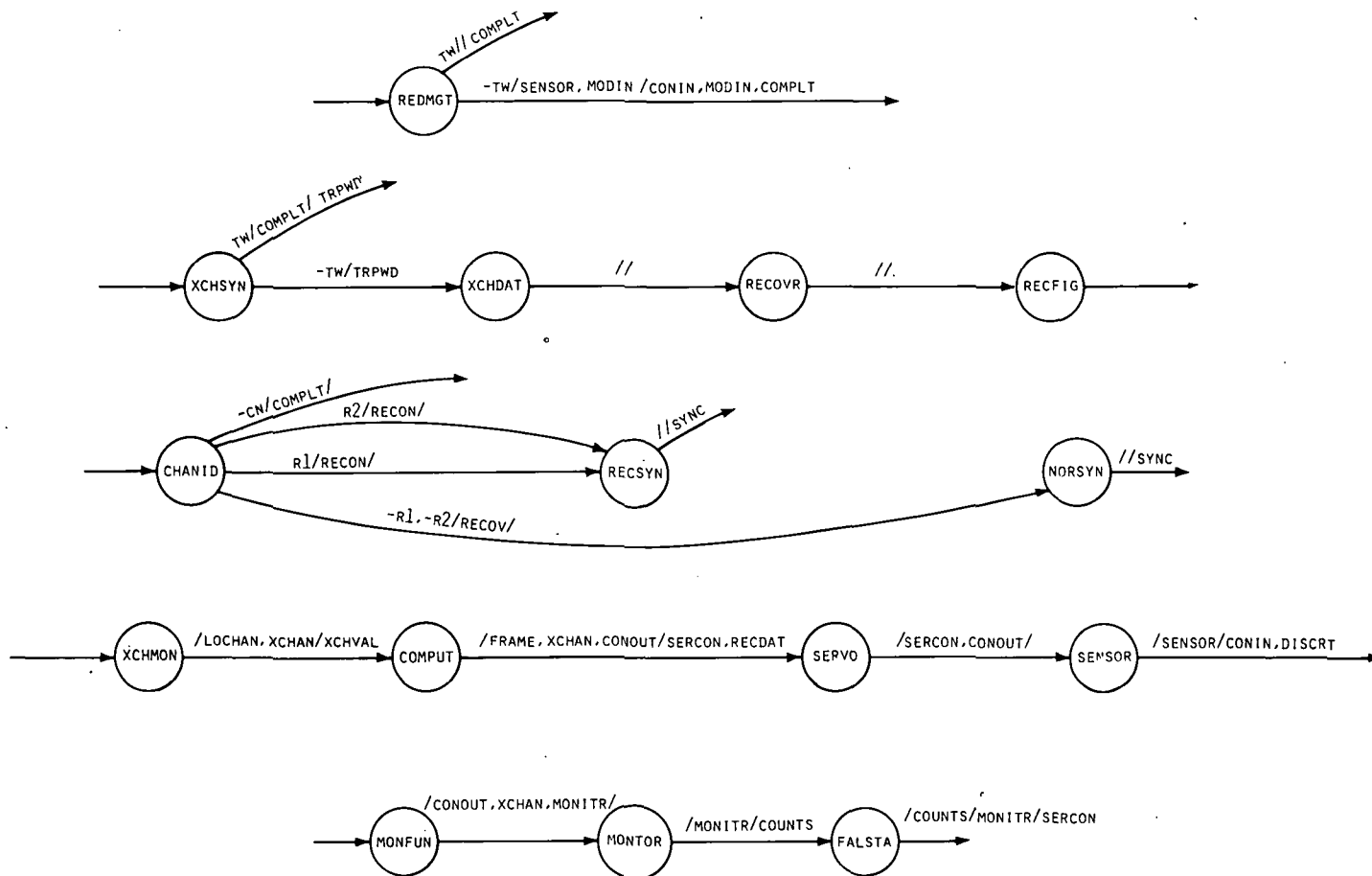


FIGURE B3-A ARCS TRANSITION DIAGRAM



50%

FIGURE B3-B REDUNDANCY MANAGEMENT TRANSITION DIAGRAM.

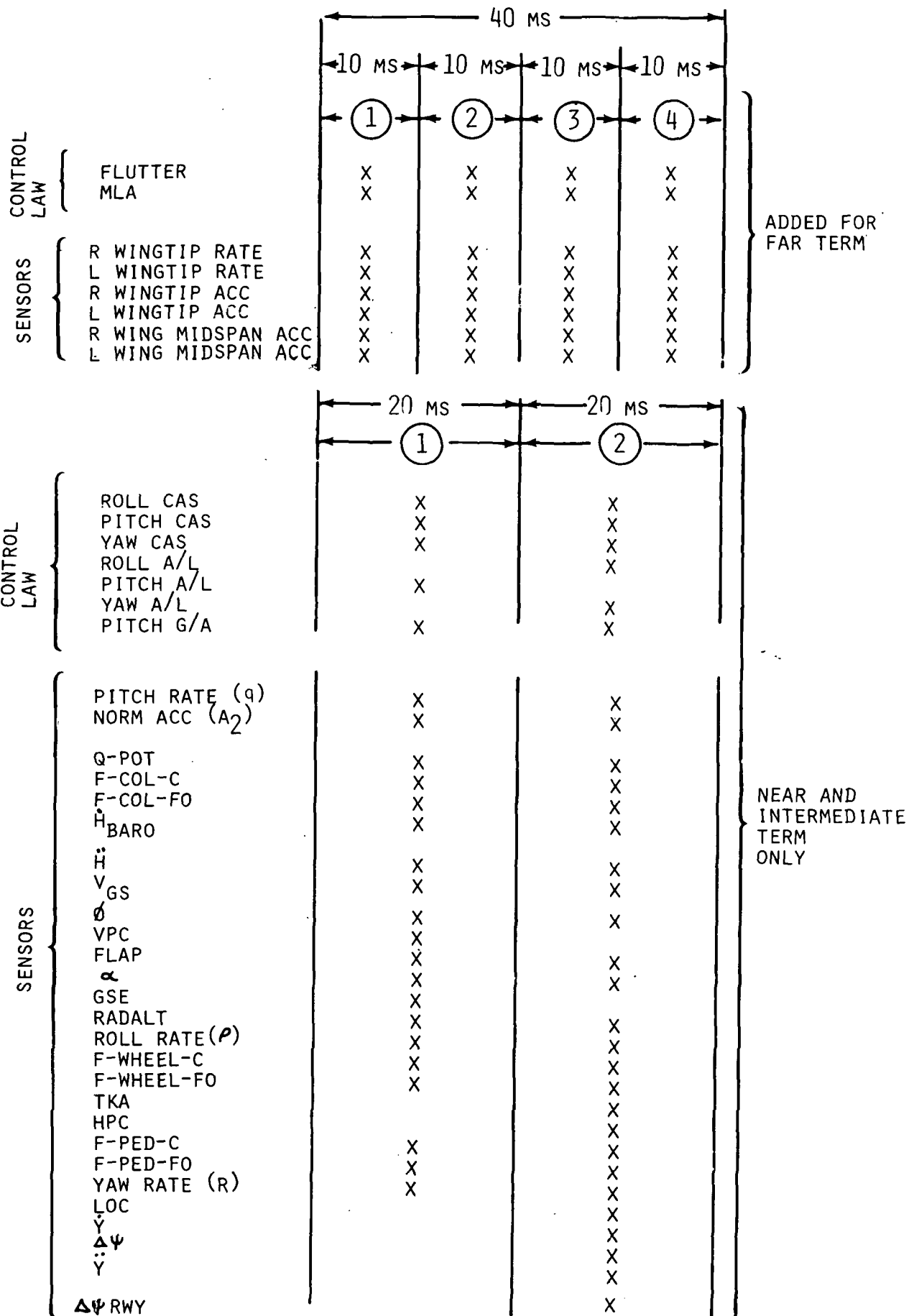


FIGURE B4 SCHEDULING TABLE

The structured name for each module can be used to form an index to the module descriptions. This index is given in Table B1.

TABLE B1: SOFTWARE INDEX	<u>Page</u>
<u>Structured Name</u>	
ARCS	
ARCS.RELTIM	
.INTPRO	
.ITREST	
.MEMPAR	
.ARIOVR	
.POWRDN	
.INOUT	
.FORGND	
.FORGND.REDMGT	
.MODCON	
.CONLAW	
.REDMGT.XCHSYN	
.XCHDAT	
.RECOVR	
RECFIG	
.RECFIG.SENSOR	
.COMPUT	
.XCHMON	
.SERVO	
.COMPUT.MONFUN	
.MONTRP	
.FALSTA	
.BAKGND	
.ONLINE	
.MANDAT	
.STPRO	
ARCS.GNDTST	
.PREMAN	
.INITIAL	
.TEST	
.DISPLY	

Module: ARCS; Advanced Reconfigurable Computer System Software

a) Description: The ARCS software is broken into real-time and non-real time operations. The real-time operations are started upon the occurrence of a power-on interrupt. Non-real time operation is started upon a request from the system test panel for ground test and verification that the aircraft is on the ground. Control is passed back to the real time operation when ground test is complete or the aircraft starts moving. Recovery must be requested upon power-on or ground test complete.

b) Sub-Processes:

RELTIM — Real time operations.

GNDTST — Non-real time operations.

c) Inputs and Outputs of the sub-processes:

RELTIM		GNDTST	
Inputs	Outputs	Inputs	Outputs
GTRD	GDTST	GTRD	RECOV

d) Data - element definitions:

GDTST — Ground test request allowed.

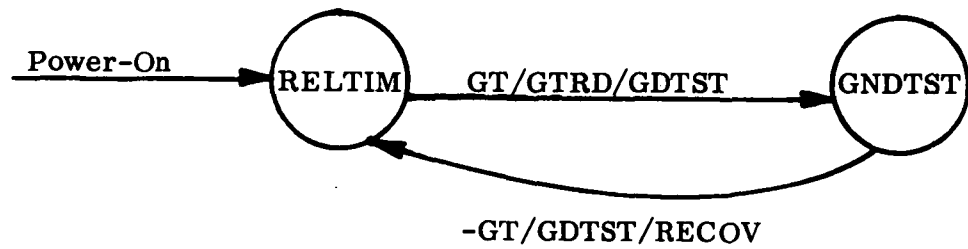
GTRD — Ground test request discrete.

RECOV — Recovery required indicator.

- 0 — Recovery not needed
- 1 — Power on recovery
- 2 — Watchdog monitor recovery
- 3 — Recover left channel
- 4 — Recover right channel

Module: ARCS; (cont'd)

e) Transition Diagram:



f) Conditions

GT — The ground test condition.

True — ground test requested and the aircraft is stationary.

False — otherwise.

Module: ARCS, RELTIM; Real Time Operations

a) Description: The real time operation is composed of interrupt processing, foreground tasks, and background tasks. Initial entry to this module is to the interrupt processing task upon a power-on interrupt or to the foreground task where recovery will be performed. When the foreground computation is complete, background is entered and continues until an iteration reset causes a transfer to interrupt processing or ground test is requested and allowed. If the foreground task is not completed before the iteration reset interrupt occurs, then a computation not complete flag is set.

b) Sub-processes:

FORGND — Foreground tasks

BAKGND — Background tasks

INTPRO — Interrupt processing

c) Inputs and Outputs of the sub-processes

FORGND		BAKGND		INTPRO	
Inputs	Outputs	Inputs	Outputs	Inputs	Outputs
RECOV	COMPLT	GTRD	GDTST	INT	INTRET
WATMON		INTRET			RECOV

d) Data-element definitions:

COMPLT — Computation complete flag.

0 — Not complete

1 — Complete

GDTST — Ground test request allowed.

GTRD — Ground test request discrete.

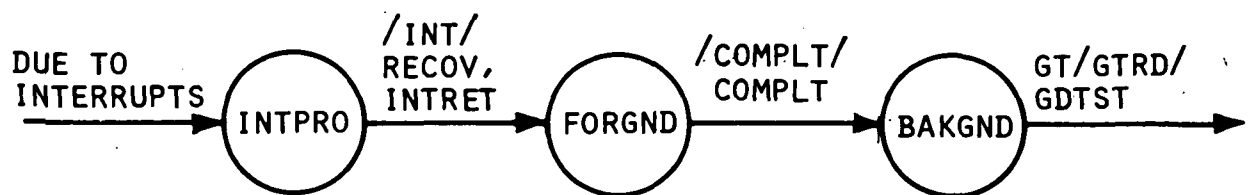
INT — Interrupt.

INTRET — Interrupt return information.

RECOV — Recovery required flag.

WATMON — Watchdog monitor flag.

Module: ARCS. RELTIM; (cont'd)



f) Conditions:

GT — Ground test.

Module: ARCS, RELTIM, INTPRO; Interrupt Processing

- a) Description: The interrupt processing task consists of interrupt handling for the iteration reset/recovery, memory parity, arithmetic overflow, and power-on interrupts. The power-on and iteration reset/recovery interrupts cause a transfer to the foreground task with the recovery request flag set to the appropriate value. The memory parity and arithmetic overflow interrupts cause their occurrence to be recorded and control is returned to the point that the interrupt occurred. The information that one of these interrupts occurred is used to localize a fault given that an output monitor tripped.

- b) Sub-processes:

ITREST — Iteration reset/recovery
 MEMPAR — Memory parity
 ARIOVR — Arithmetic overflow
 POWRON — Power-On
 INOUT — Input/Output

- c) Inputs and Outputs of the sub-processes:

ITREST		MEMPAR		INOUT	
Input	Output	Input	Output	Input	Output
INT	INTRET RECOV WATMON	INT	INTMP	INT	To be defined.

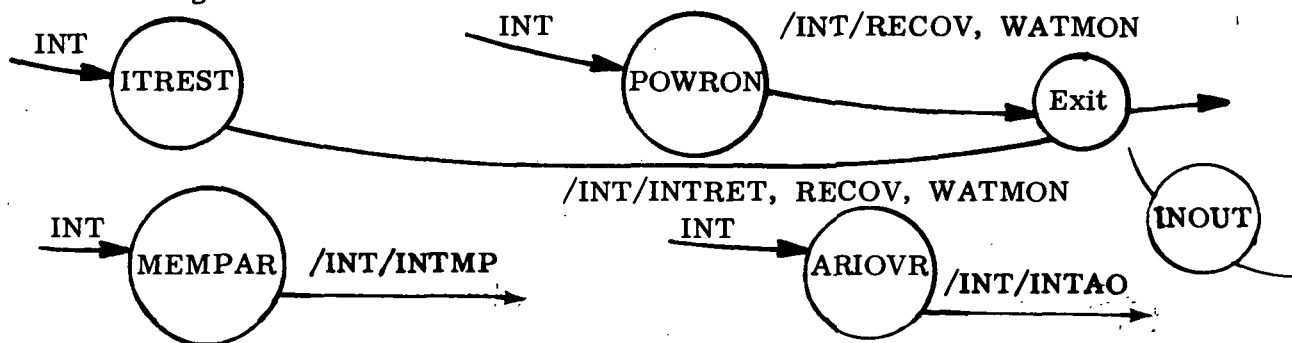
ARIOVR		POWRON	
Input	Output	Input	Output
INT	INTAO	INT	RECOV WATMON INTRET

- d) Data-element definitions:

INT — Interrupt
 INTAO — Record of frame that last arithmetic overflow interrupt occurred.
 INTMP — Record of frame that last memory parity interrupt occurred.
 INTRET — Interrupt return information
 RECOV — Recovery request flag
 WATMON — Watchdog monitor status flag.

Module: ARCS. RELTIM. INTPRO.; (cont'd)

e) Transition Diagram:



Module: ARCS. RELTIM. INTPRO. ITREST; Iteration Reset

- a) Description: The iteration reset interrupt initiates a new frame. There are two cases that must be considered depending on the status of the watchdog monitor.

The first case is when the watchdog monitor is tripped. In this case the recovery flag is set to zero to indicate normal operation if a power on recovery is shown. Otherwise, the recovery flag is set to two to request watchdog monitor trip recovery and the RAM is initialized.

The second case is when the watchdog monitor is not tripped. In this case the recovery indicator is set to one to indicate that a latent failure of the watchdog has been detected, if the recovery indicator is showing power-on recovery. Otherwise, the recovery flag is set to zero to indicate normal operation.

Module: ARCS. RELTIM. INTPRO. MEMPAR; Memory Parity

- a) Description: The memory parity interrupt indicates an error in the parity of a computer word. When this interrupt occurs, the occurrence will be recorded by the local channel and control returned to the point of interrupt.

Module: ARCS. RELTIM. INTPRO. ARIOVR; Arithmetic Overflow

- a) Description: An arithmetic overflow will be handled by limiting the affected register to the maximum allowable, positive or negative value. The overflow will be recorded in the local channel and the control returned to the point the process was interrupted.

Module: ARCS. RELTIM. INTPRO. POWRON; Power On

- a) Description: The power-on interrupt will force computation to begin with this function. The computer's RAM will be initialized and a power-on recovery will be requested by setting the recovery flag to one.

Module: ARCS. RELTIM. INTRPO.I/OINT: Input/Output Interrupt

a) Description: For purposes of the ARCS Concept the "stack empty" interrupt will cause the following action to be taken. The FIFO stack will be filled with recovery data and transmitted. After all recovery data has been transmitted

the interrupt will be masked. For the near term model approximately six buffer loads of recovery data will need to be transmitted each frame. This will require about 1 ms of CPU time per minor frame to load the buffer. For the complete transfer of data to the foreign channels it will take 8 ms.

Module: ARCS. RELTIM. FORGND; Foreground (Synchronous) Tasks

- a) Description: The foreground tasks are redundancy management, control laws, and mode control which are processed in that order. Redundancy management among other functions produces inputs for the mode control and control law processes. Mode control determines the flight mode based on pilot selection and flight regime. The control law computes the servo commands and sets certain flight regime indicators.

A determination is made in redundancy management whether to trip the watchdog monitor. This determination is made based on system status as recorded in the system status table.

- b) Sub-processes:

REDMGT — Redundancy management
CONLAW — Control law
MODCON — Mode control

- c) Inputs and Outputs of the sub-processes:

REDMGT	
Inputs	Outputs
SENSOR	CONIN
DISCRT	MODIN
COMPLT	TRPWD

CONLAW	
Inputs	Outputs
MODE	COMPLT
CONIN	CONOUT
	LAWDIS

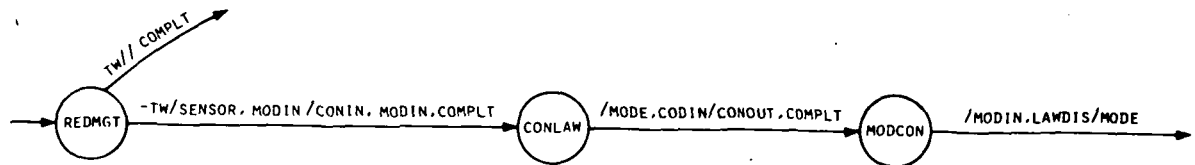
MODCON	
Inputs	Outputs
MODIN	MODE
LAWDIS	

- d) Data-element definitions:

COMPLT — Computation Complete
CONIN — Control law inputs
CONOUT — Control law outputs
MODIN — Mode Control Inputs
LAWDIS — Control law computed flags
MODE — Selected flight mode
SENSOR — Sensor values

Module: ARCS. RELTIM. FORGND; (cont'd)

e) Transition Diagram:



f) Conditions:

TW — Watchdog monitor tripped.

B.1.2

ARCS Redundancy Management Software Modules

An important aspect of the executive function associated with the Redundancy Management operation is that tied to the reconfiguration process. The reconfiguration process encompasses those functions required to assess the system fault status. Figure B5 is a breakdown of the system status indicators. This breakdown shows the hierarchical data structure of the system status table which drives the reconfiguration process. Based on the information contained in this table, the local computer can determine the system redundancy level. The recognized system redundancy level, based on synchronization data, will in turn set portions of the system status table to effect redundancy management processing which is consistent with the redundancy level up/down transitions.

The following is a module-by-module description of the Redundancy Management software.

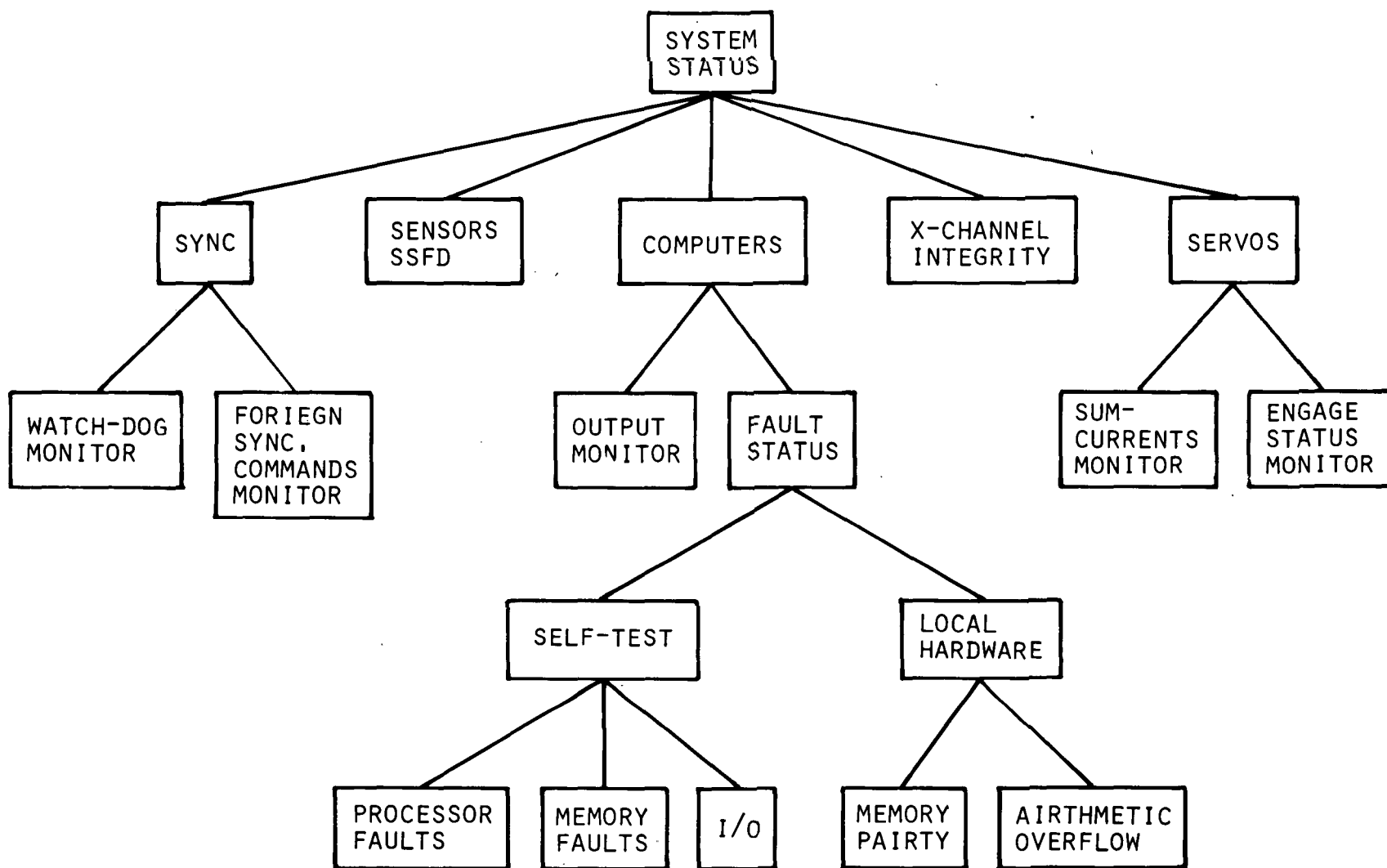


FIGURE B5 TREE BREAKDOWN OF SYSTEM STATUS

- a) Description: Redundancy management manages processes that are concerned with multiple copies of the same information or the alignment in time of redundant processes. Specifically, redundancy management consists of reconfiguration, cross channel synchronization, recovery and cross channel data transfer.

The synchronization process produces synchronization status, channel identity, and software minor frame count. This information is used by the reconfiguration process in performing its monitoring function. The software frame count is used by the cross channel transfer process in selecting the information to be transmitted.

The reconfiguration process may perform recovery based on its monitoring. In this event the recovery process would use the cross channel data that was received from one of the foreign channels.

- b) Sub-processes:

XCHSYN — Cross channel synchronization
 XCHDAT — Cross channel data transfer
 RECFIG — Reconfiguration
 RECOVR — Recovery processes

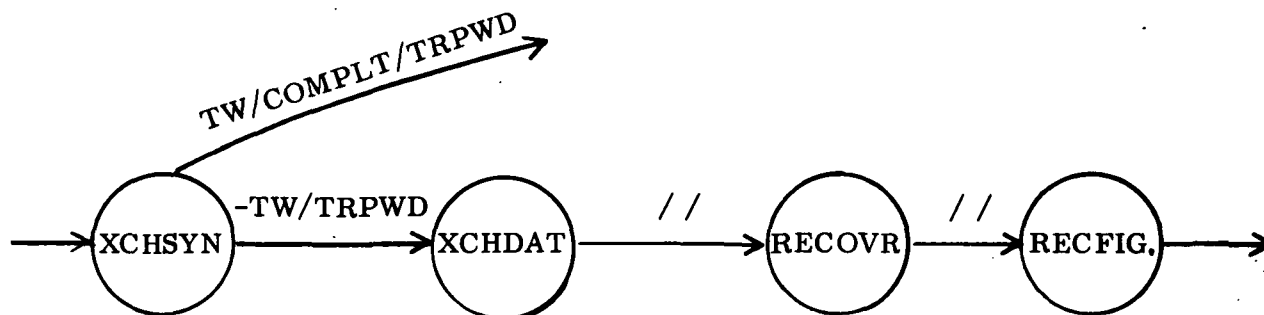
- c) Inputs and Outputs of the Sub-processes:

XCHSYN		XCHDAT		RECFIG	
Inputs	Outputs	Inputs	Outputs	Inputs	Outputs
RECOV	FRAME	FRAME	To other channels	FRAME	CONIN
WATMON	LOCHAN	SVEC		LOCHAN	DISCRT
COMPLT	SYNC	CONOUT		SYNC	RECDAT
	COMPLT	MODE		CONOUT	
	TRPWD				
RECOVR					
Inputs	Outputs				
XCHDT	SVEC				
SYNC	FRAME				
RECOV					

d) Data-Element Definitions:

COMPLT — Computation Complete
RECOV — Recovery request flag
WATMON — Watchdog monitor status flag
FRAME — Local minor frame count
LOCHAN — Local channel identity
SYNC — Synchronization status word
CONOUT — Control law output
CONIN — Control law input
DISCRT — Discretes
RECDAT — Recovery request for data
XCHDT — Cross channel data for foreign channels
SVEC — State vector
TRPWD — Trip watchdog monitor

e) Transition Diagram:



f) Conditions:

TW — Trip watchdog monitor

- a) Description: The cross channel synchronization process consists of a recovery, channel identity and a normal synchronization process. First, channel identity is determined then based on the watchdog monitor being tripped or recovery being requested recovery synchronization is performed. If neither of those conditions are true, then normal synchronization is performed, unless the previous frames computation was not completed. When computation is not completed, the watchdog monitor will be tripped by not setting and resetting the local sync commands.

- b) Sub-processes:

CHANID — Channel identity word

RECSYN — Recovery synchronization

NORSYN — Normal synchronization

- c) Inputs and Outputs of the Sub-processes:

CHANID		RECSYN		NORSYN	
Inputs	Outputs	Inputs	Outputs	Inputs	Outputs
COMPLT					
LA	LOCHAN	LA	SYNC	LA	SYNC
LB	TRPWD	LB	COMPLT	LB	COMPLT
WATMON					
RECOV					

- d) Data-Element Definitions:

COMPLT — Computation complete

LA and LB — These indicators identify the local channel.

LOCHAN — Identity of the local channel

FRAME — Local minor frame count

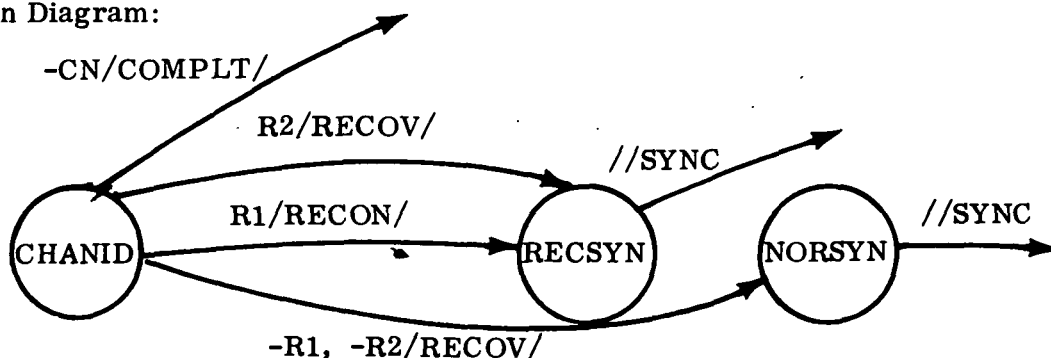
RECOV — Recovery request flag

SYNC — Synchronization status word

TRPWD — Trip watchdog monitor

WATMON — Watchdog monitor status flag

- e) Transition Diagram:



- f) Conditions:

CN — Computation complete

R2 — Recovery required (RECOV is 2)

R1 — Recovery required (RECOV is 1)

- a) Description: The channel identification process identifies the local, left, and right channels. This is accomplished by checking the indicators LA and LB shown in Table B2 and using this identity in Table B3 to establish local, left and right.

LA	LB	Channel
0	0	A
0	1	—
1	0	B
1	1	C

Table B2: CHANNEL IDENTIFICATION

If Local is	Then Left is	Then Right is
A	C	B
B	A	C
C	B	A

Table B3: LEFT AND RIGHT CHANNEL IDENTIFICATION

b) Subprocesses:

CHANA — Local channel is A
 CHANB — Local channel is B
 CHANC — Local channel is C

c) Inputs and Outputs of the sub-processes:

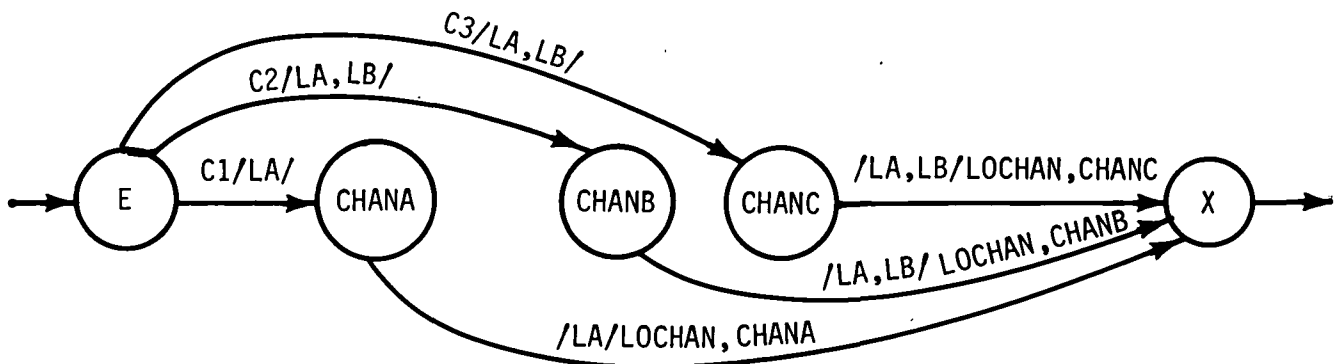
CHANA	
Inputs	Outputs
LA	LOCHAN
LB	LCHAN
	RCHAN

CHANB	
Inputs	Outputs
LA	LOCHAN
LB	LCHAN
	RCHAN

d) Data-Element Definitions:

LA —
 LB —
 LOCHAN — Local channel
 LCHAN — Left channel
 RCHAN — Right channel

e) Transition Diagram:



f) Conditions:

C1 — LA is zero
 C2 — LA is one and LB is zero
 C3 — LA is one and LB is one

Module: ARCS. RELTIM. FORGND. REDMGT. XCHSYN. CHANID. CHANA;
Local Channel is A

- a) Description: The variable LOCHAN is set to zero, LCHAN is set to two, and RCHAN is set to one.

Module: ARCS. RELTIM. FORGND. REDMGT. XCHSYN. CHANID. CHANB;
Local Channel is B

- a) Description: The variable LOCHAN is set to one, LCHAN is set to zero, and RCHAN is set to two.

Module: ARCS. RELTIM. FORGND. REDMGT. XCHSYN. CHAND. CHANC;
Local Channel is C

- a) Description: The variable LOCHAN is set to three, LCHAN is set to one, and RCHAN is set to two.

Module: ARCS. RELTIM. FORGND. REDMGT. XCHSYN. RECSYN;
Recovery Synchronization

- a) Description: The initial synchronization algorithm was described in Section 2.1.1.

Module: ARCS. RELTIM. FORGND. REDMGT. XCHSYN. NORSYN;
Normal Synchronization

- a) Description: The normal synchronization algorithm was described in Section 2.1.1. A recovery indicator will be set by this process to indicate when the local computer first synchronizes with another computer. This indicator will be used to release permanent failure flags for the recovering computer.

Module: ARCS.RELTIM.FORGND.REDMGT.XCHDAT: Cross Channel Transmission

a) Description: This process will initiate cross channel transmission by unmasking the buffer empty interrupt and transmitting that information that is needed for this frame's computation and redundancy management.

b) Sub-processes:

W — Wait for transmission complete
 UM — Unmask interrupt
 SWT — Software frame to FIFO
 SNT — Sensor Data to FIFO
 SRT — Servo Data to FIFO

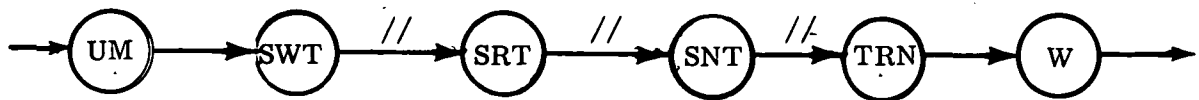
c) Inputs and outputs of the sub-processes:

UM	SWT		SNT		SRT	
None	Inputs	Outputs	Inputs	Outputs	Inputs	Outputs
	FC	to FIFO	Sensor Input for Frame FC	TO FIFO	Servo Commands for Frame FC	TO FIFO

d) Data-Element Definitions

FC — Frame Count
 FIFO — Buffer for cross channel transmission.

e) The Transition Diagram of



Module: ARCS, RELTIM, FORGND, REDMGT, RECOVER; Recovery

a) Description: The recovery process consists of resetting permanent failure flags in a working computer when it synchronizes with another and then proceeding with its normal computations. Recovery for the faulted computer consists of updating its flags and history to a working computer; and then proceeding with normal computation.

b) Sub-processes:

UPDATE — Update flags and history

RESETL — Reset failure flags for the left computer.

RESETR — Reset failure flags for the right computer.

c) Inputs and Outputs of the sub-processes:

UPDATE	
Inputs	Outputs
LSYNC	FRAME
RSYNC	SYSTAB
	HISTORY
LRCVR	
RRCVR	

RESETL	
Inputs	Outputs
LSYNC	— LFFLAG

RESETR	
Inputs	Outputs
RSYNC	RFFLAG

d) Data-Elements:

LSYNC — Left sync flag

RSYNC — Right sync flag

LRCVR — Left receiver

RRCVR — Right receiver

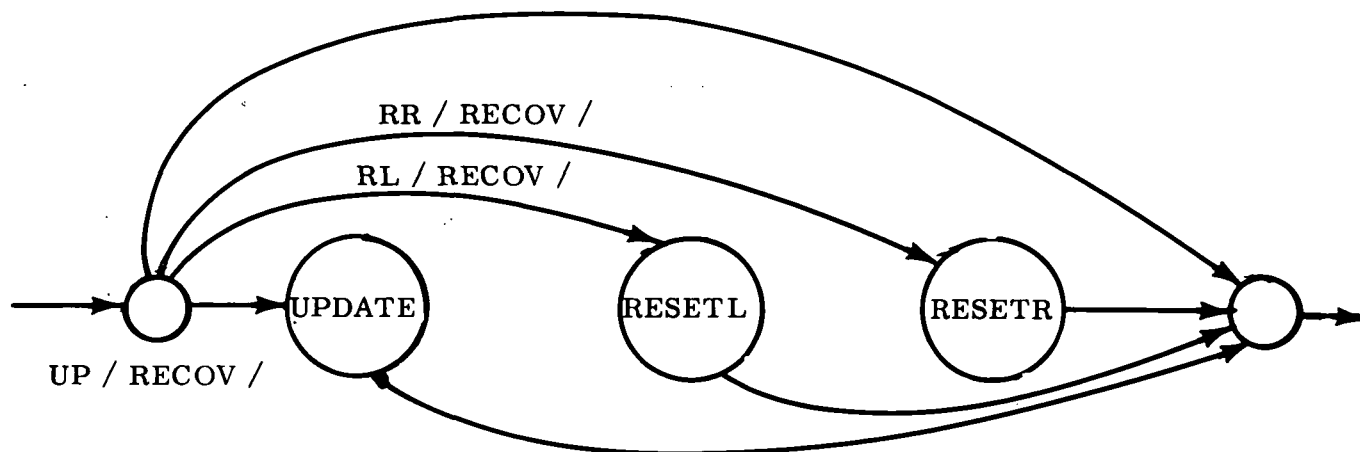
FRAME — Local frame count

HISTORY — Local history

LFFLAG — Left failure flags

RFFLAG — Right failure flags

e) Transition Diagram: NR / RECOV /



f) Conditions:

- NR — No recovery (RECOV is 0)
- UP — Update local computer (RECOV is 1 or 2)
- RL — Recover the left computer (RECOV is 3)
- RR — Recover the right computer (RECOV is 4)

Module: ARCS. RELTIM. FORGND. REDMGT. RECOVER. UPDATE: Update Local
Information

- a) Description: Based on the synchronization status the local computer will either update its local information to the left or right channel. If the local computer is not synchronized with another computer then it will start processing and buildup its own history over a period of frames.

The flow chart of Figure B6 shows the flow of control for this process.

Each block of recovery data that is received by the recovering computer will have a code word at the end of it. This word will indicate that new data has been received. The word will be checked until it indicates new data received, then this new data will be used to update the recovering computer's history.

At timing loop could be used here to assure that a cross-channel data fault does not prevent the recovering computer from getting into operation.

The strategy would be to try the left, if that does not work try the right. If neither the left or right is sending data, then in the presence of no other local faults begin simplex operation.

Module: ARCS. RELTIM. FORGND. REDMGT. RECOVER. RESETL;
Reset Left

- a) Description: The permanent failure flags for the left computer are reset. The function is performed when the local working computer sees a previously faulted computer come into synchronization.

Module: ARCS. RELTIM. FORGND. REDMGT. RECOVER. RESETR;
Reset Right

- a) Description: The permanent failure flags for the right computer are reset. Again, this is performed by a local working computer when a previously failed computer comes into synchronization.

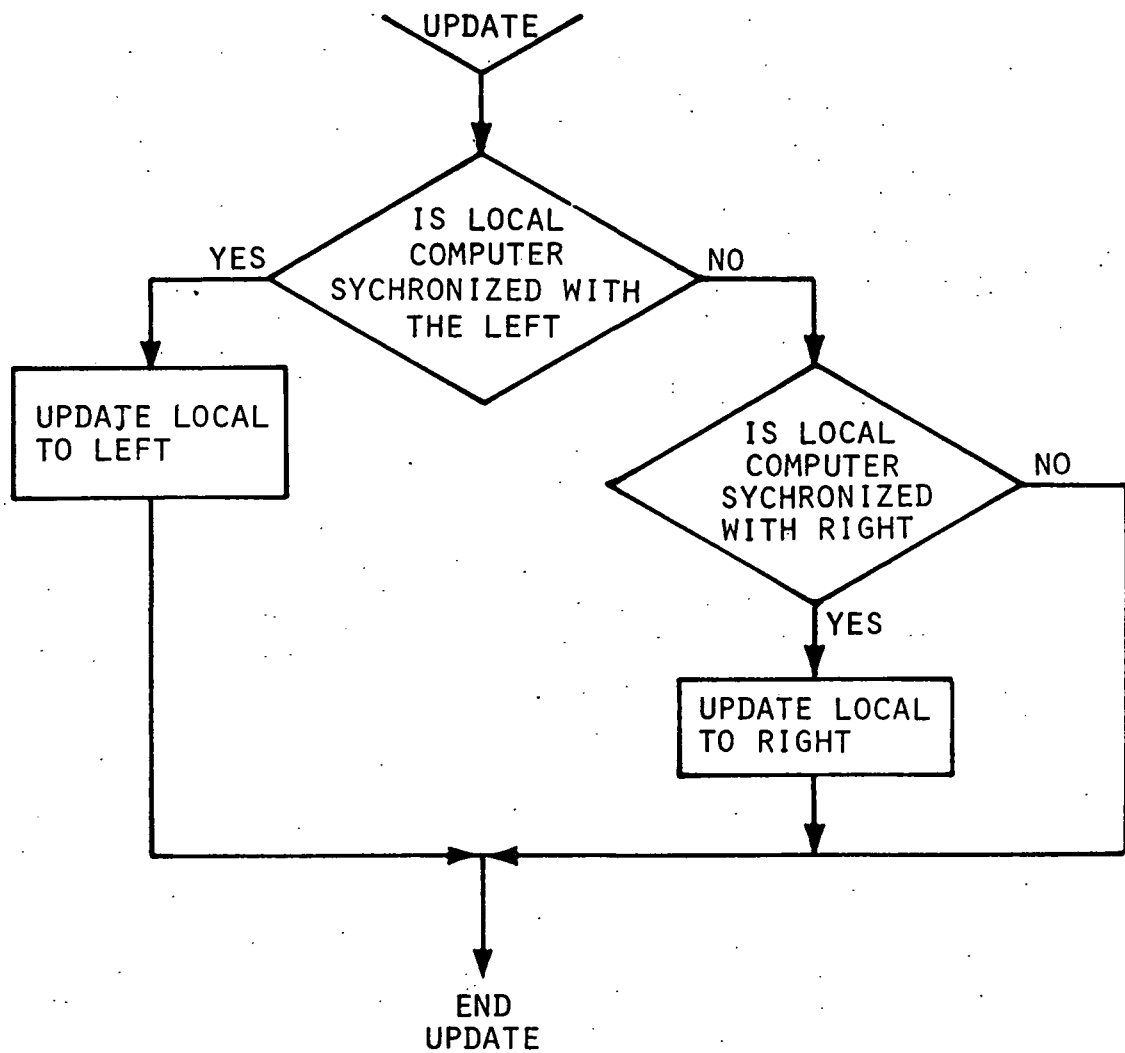


FIGURE B6 UPDATE FUNCTION

Module: ARCS. RELTIM. FORGND. REDMGT. RECFIG; Reconfiguration

- a) Description: The reconfiguration process is concerned with sensors, computers, cross channel data links, and servos. The process is based on the monitors for these four pieces of hardware and the results of the synchronization process. The synchronization information is used to establish the overall redundancy level as seen by the local channel. Monitoring of the various hardware components is used to obtain the redundancy level of those components. This is presented in detail when the design for the individual components is discussed.

The cross channel is monitored first to establish the validity of the cross channel information. Second, the computer monitoring is performed to assure software frame synchronization and the servo commands that were computed last frame. Third, the servo process is performed. Finally, the sensor selection failure detection is performed which starts a new set of computations that will be monitored next frame.

- b) Sub-processes:

SENSOR — Sensor select failure detect.

COMPUT — Computer monitoring

XCHMON — Cross channel monitor

SERVO — Servo monitor and control

- c) Inputs and Outputs of the Sub-processes:

SENSOR	
Inputs	Outputs
from the sensors XCHAN	CONIN PISCRT

COMPUT	
Inputs	Outputs
CONOUT XCHAN FRAME	SERCON

XCHMON	
Inputs	Outputs
XCHAN LOCHAN	XCHVAL

SERVO	
Inputs	Outputs
SERCON CONOUT	To the servos

d) Data-Element Definitions:

XCHAN — Cross channel information that was received.

CONIN — Control law inputs

MODIN — Mode Control Inputs

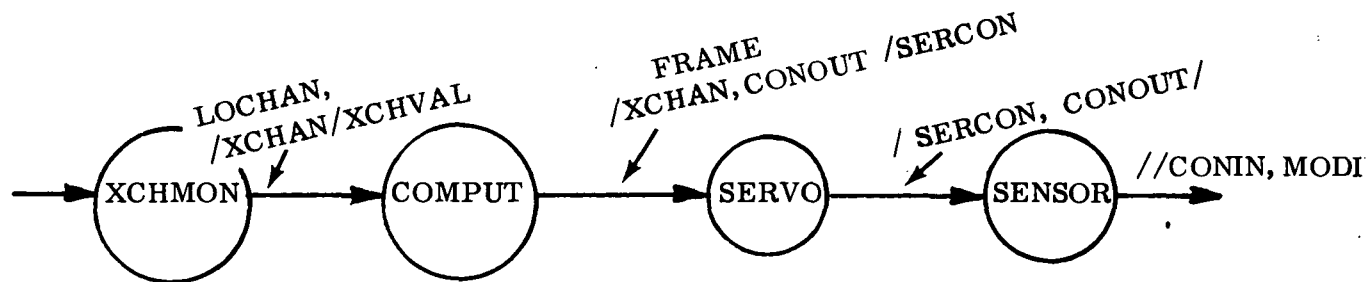
FALSTA — Local fault status

CONOUT — Control law outputs

SERCON — Servo control flag consisting of monitors and permanent flags.

XCHVAL — Cross channel valid word.

e) Transition Diagram:



Module: ARCS, RELTIM, FORGND, REDMGT, RECFIG, SENSOR, Sensors

- a) Description: The basic function of this module is to provide the control law process with necessary inputs based on sensor values from all operating channels. In a given minor frame a number of sensor selection failure detection (SSFD) processes will operate based on the minor frame count, as shown in the following paragraph.

The particular sensor sets that must be processed in a given frame can be derived from the scheduling table of **Figure 2-20**.

- b) Sub-processes:

SSFO	— Sensor Select Failure Detect for frame 0	} Near Term	} Far Term	
SSF1	— Sensor Select Failure Detect for frame 1			
SSF2	— Sensor Select Failure Detect for frame 2	}		
SSF3	— Sensor Select Failure Detect for frame 3			

- c) Inputs and Outputs to the Sub-processes:

SSF _i	
Inputs	Outputs
SENSOR	CONIN
SYSTAB	SYSTAB

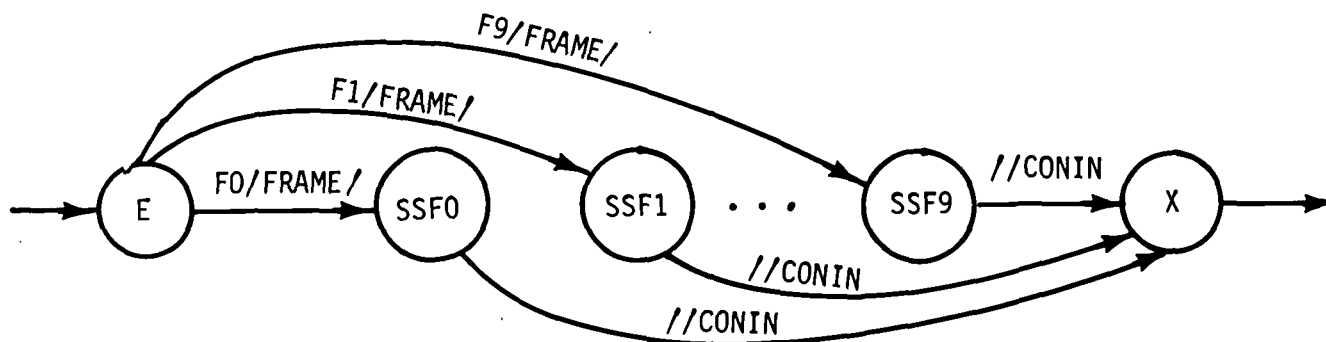
- d) Data-Element Definitions:

SENSOR — Sensor values

SYSTAB — System status table. The synchronization information will be used as an input and the sensor information will be updated.

CONIN — Control law inputs.

e) Transition Diagram:



N is 1 for the near term

N is 2 for the far term

f) Conditions:

F0 — FRAME is 0.

F1 — FRAME is 1.

.

FN — FRAME N.

Module: ARCS. RELTIM. FORGND. REDMGT. RECFIG. SENSOR. SSFD,
Sensor Select Failure Detection

a) Description: The SSFD function is used for each sensor set.

For a typical SSFD the inputs are three sensor values A, B, and C, and the output is the selected sensor value, CI. For recovery purposes a number of internal variables must be available to the recovery process.

Figure B7 shows the functional breakdown of the SSFD function and the block diagram for the SSFD. This figure will be used in the development of the data-space and control flow.

b) Sub-processes:

AVER — Average of 3 or 2.

BIAS — Bias error calculation.

BECOMP — Bias error compensation

RECSEN — Reconfiguration based on fault detection.

c) Inputs and Outputs of the sub-processes:

AVER	
Inputs	Outputs
IUA	OUTAVE
IUB	
IUC	
APRIME	
BPRIME	
CPRIME	

BECOMP	
Inputs	Outputs
A	APRIME
B	BPRIME
C	CPRIME

BIAS	
Inputs	Outputs
OUT AVE	BERA
A	BERB
B	BERC
C	

RECSEN	
Inputs	Outputs
BERA	IUA
BERB	IUB
BERC	IUC
	APFRMF
APRIME	BPFRMF
BPRIME	CPFRMF
CPRIME	
OUTAVE	
A	
B	
C	

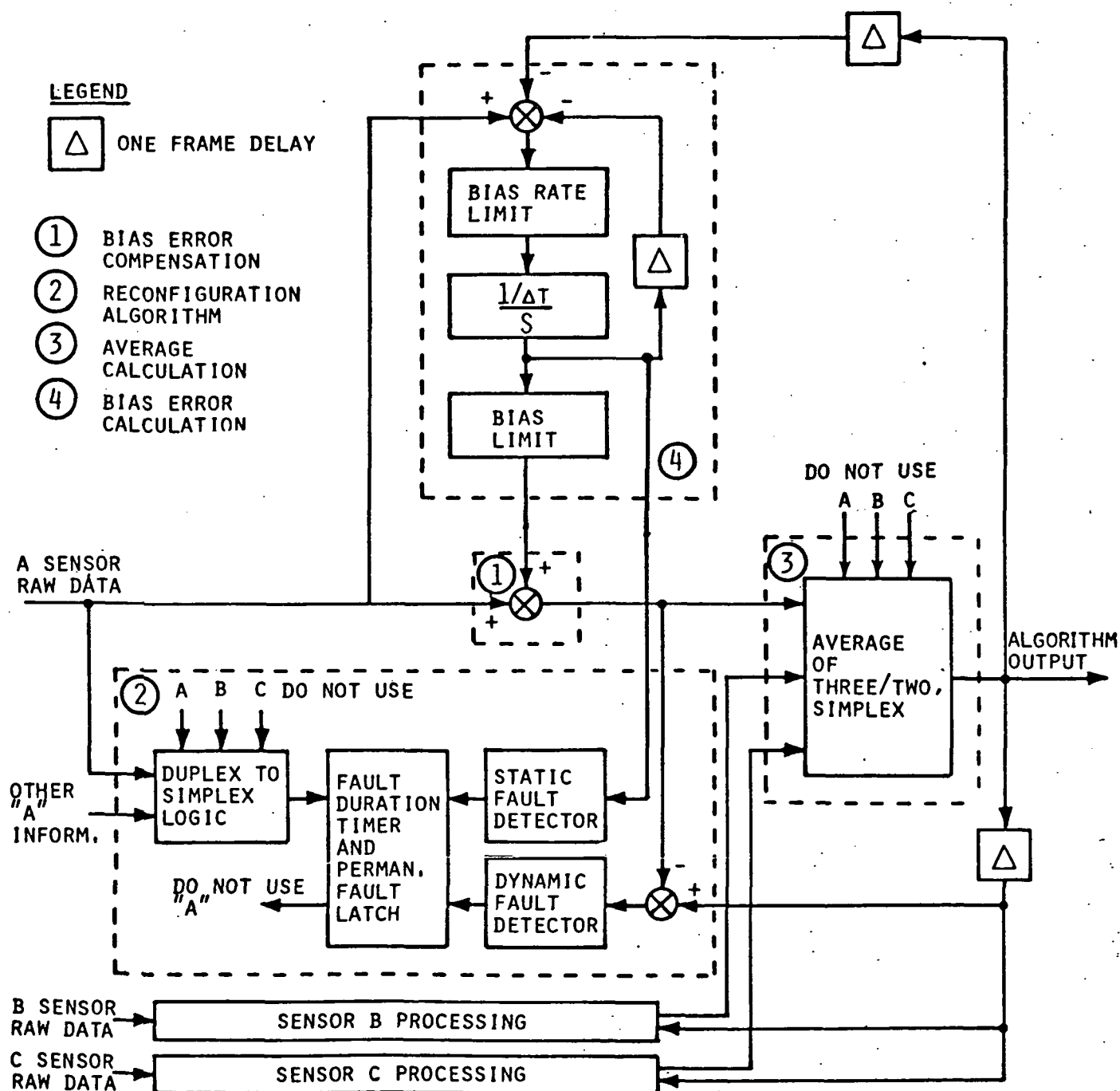
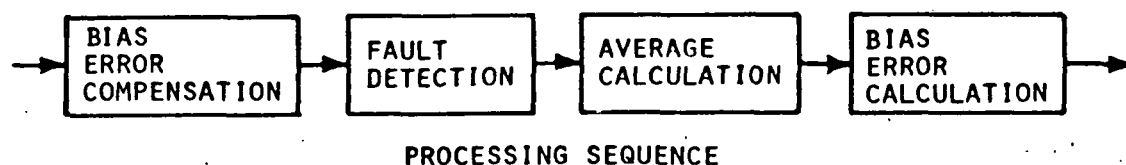


FIGURE B7. CONTINUOUS SIGNAL SELECTION/FAULT DETECTION ALGORITHM

d) Data-Element Definitions:

IU_i — Where $i = A, B, C$ Sensor i "use flag".

A value of 1 means use the sensor in the process. IU_i is from SYSTAB.

A, B, and C — The three RAW sensor values.

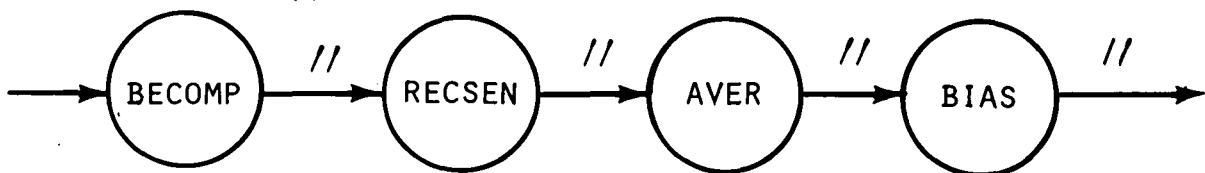
OUTAVE — The average of A, B, and C as determined by IU_i .

BER_i — $i = A, B, C$. Bias error for the sensor i .

i PRIME — $i = A, B, C$. Compensated sensor value.

i PRMF — $i = A, B, C$. Permanent failure flag which is in SYSTAB.

e) Transition Diagram:



Module: ARCS. RELTIM. FORGND. REDMGT. RECFIG. COMPUT; Computer
Monitoring and Failure Assessment

a) Description: The functions that must be accomplished by this process are output monitoring, monitor trip assessment, and failure status assessment.

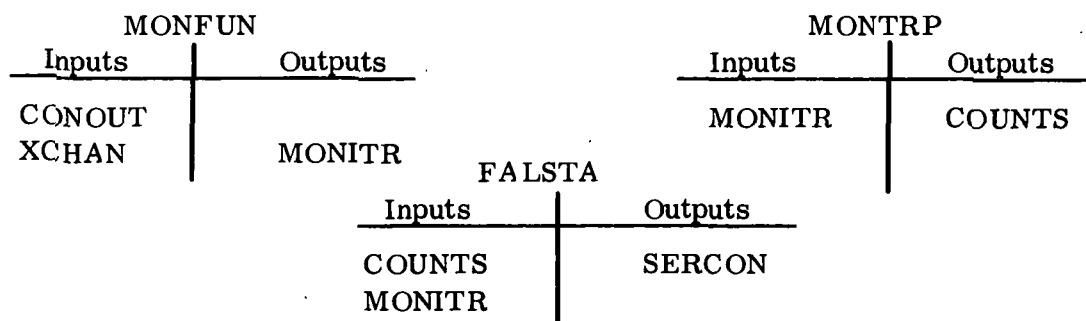
b) Sub-processes:

MONFUN — Monitor Function

MONTRP — Monitor Trip Assessment

FALSTA — Failure Status Assessment

c) Inputs and Outputs of the sub-processes:



d) Data-Element Definitions:

CONOUT — Control law outputs

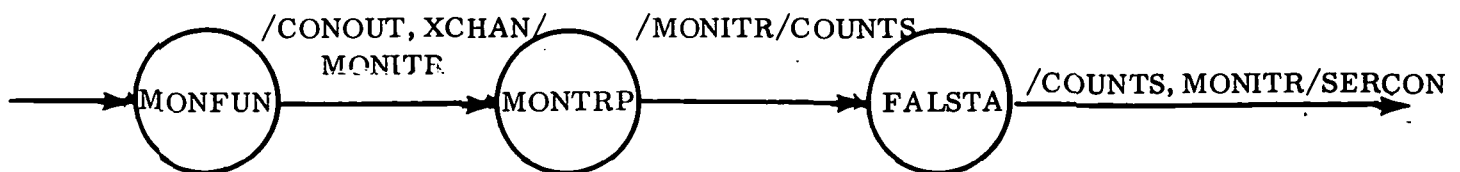
XCHAN — Cross Channel information that was received.

MONITR — Vector of the fifteen monitor flags given in Table 5.

COUNTS — Vector of the fifteen monitor trip counters defined in Table 6.

SERCON — Servo Control vector consisting of thirteen servo monitor flags and thirteen permanent isolation flags.

e)



Module: ARCS. RELTIM. FORGND. REDMGT. RECFIG. COMPUT. MONFUN;
Output Monitor Function

- a) Description: The computer monitoring process assesses the validity of the local computer's computations based on foreign computer computations. The monitoring process only makes sense for triplex and duplex operation. The primary level of redundancy is determined by the channels synchronization status and the secondary level is determined by the permanent fault flags and monitor flags.

In Table B4 the eight possible cases for the three flags are shown and the good outputs are indicated.

Case	Monitor Flags			Meaning
	A	B	C	
0	0	0	0	A, B, and C agree
1	0	0	1	A and B agree
2	0	1	0	A and C agree
3	0	1	1	Can not occur
4	1	0	0	B and C agree
5	1	0	1	Can not occur
6	1	1	0	Can not occur
7	1	1	1	A, B, and C disagree

Table B4: Monitor Flag Meaning

The monitor flags for outputs A, B, and C are set to indicate which outputs are in disagreement. The comparison shown in Figure B8 can be for equality, that the values are within some tolerance of each other, or that the signals are within tolerance of an average of all valid signals.

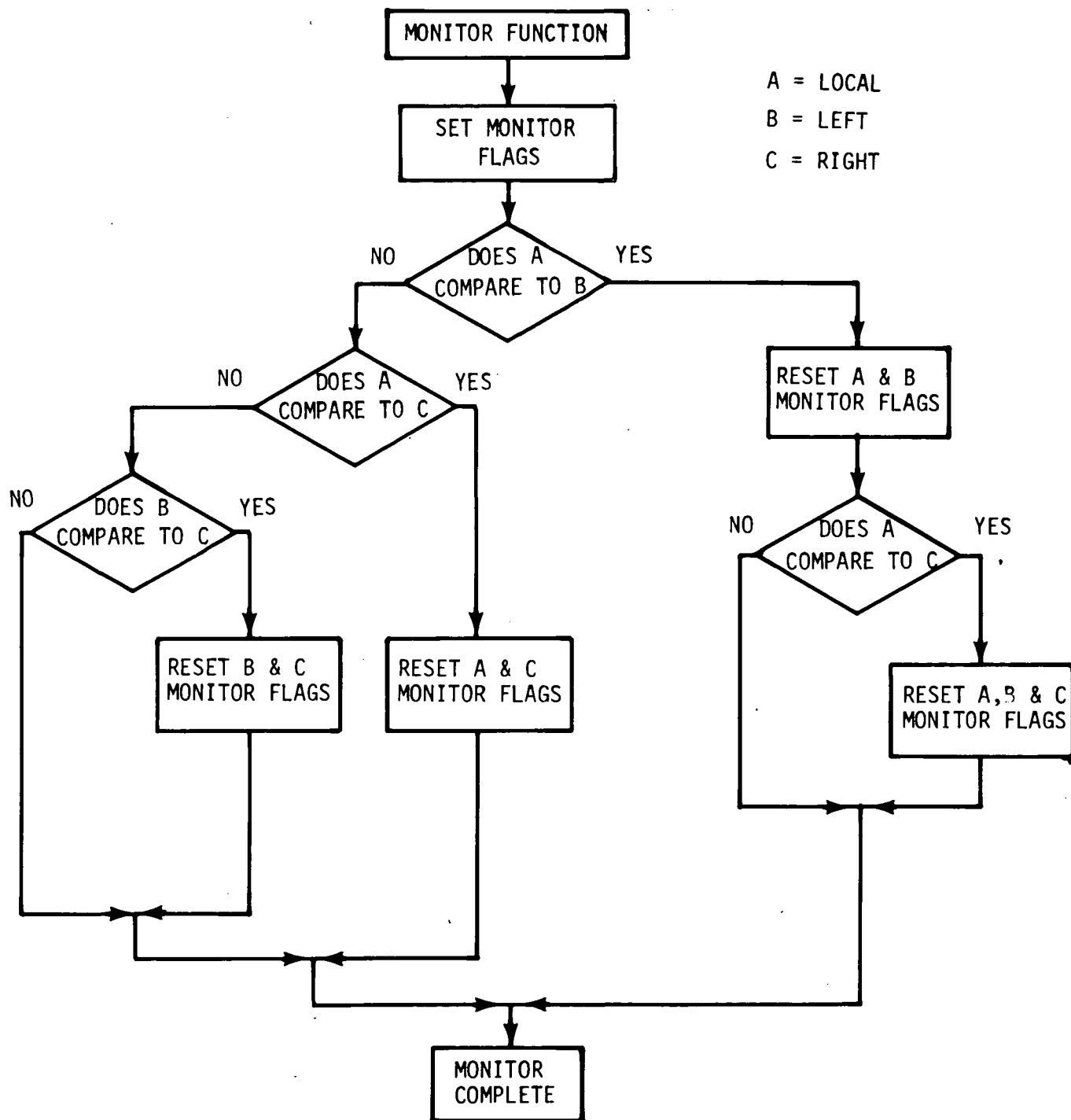


FIGURE B8 TRIPLEX MONITORING

The output monitors are listed in Table B5 with their associated monitor flags and the allowable tolerance between values.

Monitor	Tolerance	Description
MPCS		Pitch Control Surface
MUR		Upper Rudder
MLR		Lower Rudder
MLA		Left Aileron
MRA		Right Aileron
MLMS		Left Midspan Spoiler
MRMS		Right Midspan Spoiler
MLTS		Left Tip Spoiler
MRTS		Right Tip Spoiler
MLWTFC		Left Wing Tip Flutter Control
MRWTFC		Right Wing Tip Flutter Control
MLOFS		Left Outboard Flutter Suppressor
MROFS		Right Outboard Flutter Suppressor
MFRAME	Equality	Minor Frame Count
MMODE	Equality	Flight Mode

Table B5: Output Monitors

The assessment of output disagreement in duplex can not be done by monitoring alone. The monitor function for duplex is shown in figure B9. As seen from the figure either the output monitor is tripped or not tripped.

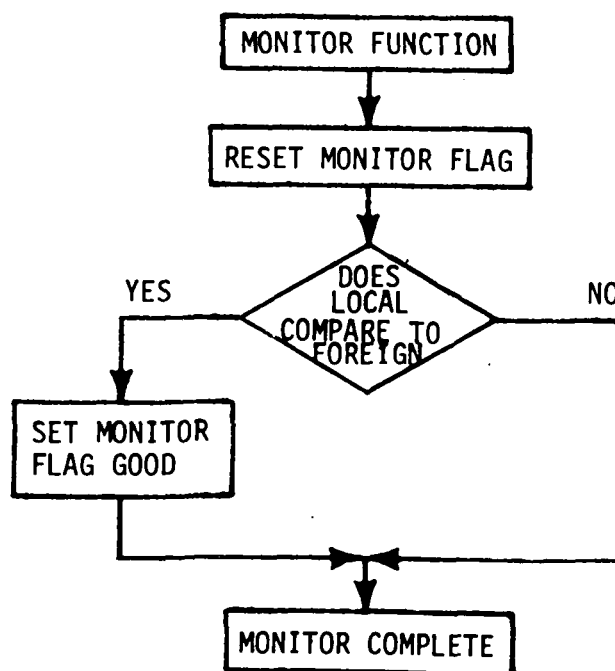


FIGURE B9 DUPLEX MONITOR

Module: ARCS. RELTIM. FORGND. REDMGT. RECFIG. COMPUT. MONTRP;
 Monitor Trip Assessment

- a) Description: The monitor trip assessment consists of counting the number of monitor trips. In general, if a monitor is tripped the counter for that monitor is incremented. The counters are given in Table B6 with the threshold at which a permanent fault will be declared.

Counter *	Threshold	Monitor Tripped
CMPCS		Pitch Control Surface
CMUR		Upper Rudder
CMLR		Lower Rudder
CMLA		Left Aileron
CMRA		Right Aileron
CMLMS		Left Midspan Spoiler
CMRMS		Right Midspan Spoiler
CMLTS		Left Tip Spoiler
CMRTS		Right Tip Spoiler
CMLWTFC		Left Wing Tip Flutter Control
CMRWTFC		Right Wing Tip Flutter Control
CMLOFS		Left Outboard Flutter Suppressor
CMROFS		Right Outboard Flutter Suppressor
CMFRAME		Minor Frame Count
CMMODE		Flight Mode

} Far
Term

* The counters are always greater than or equal to zero.

Table B6: Monitor Trip Counters

Figure B10 shows the operations that must be accomplished to perform the monitor trip assessment process in triplex. The duplex process is shown in Figure B11.

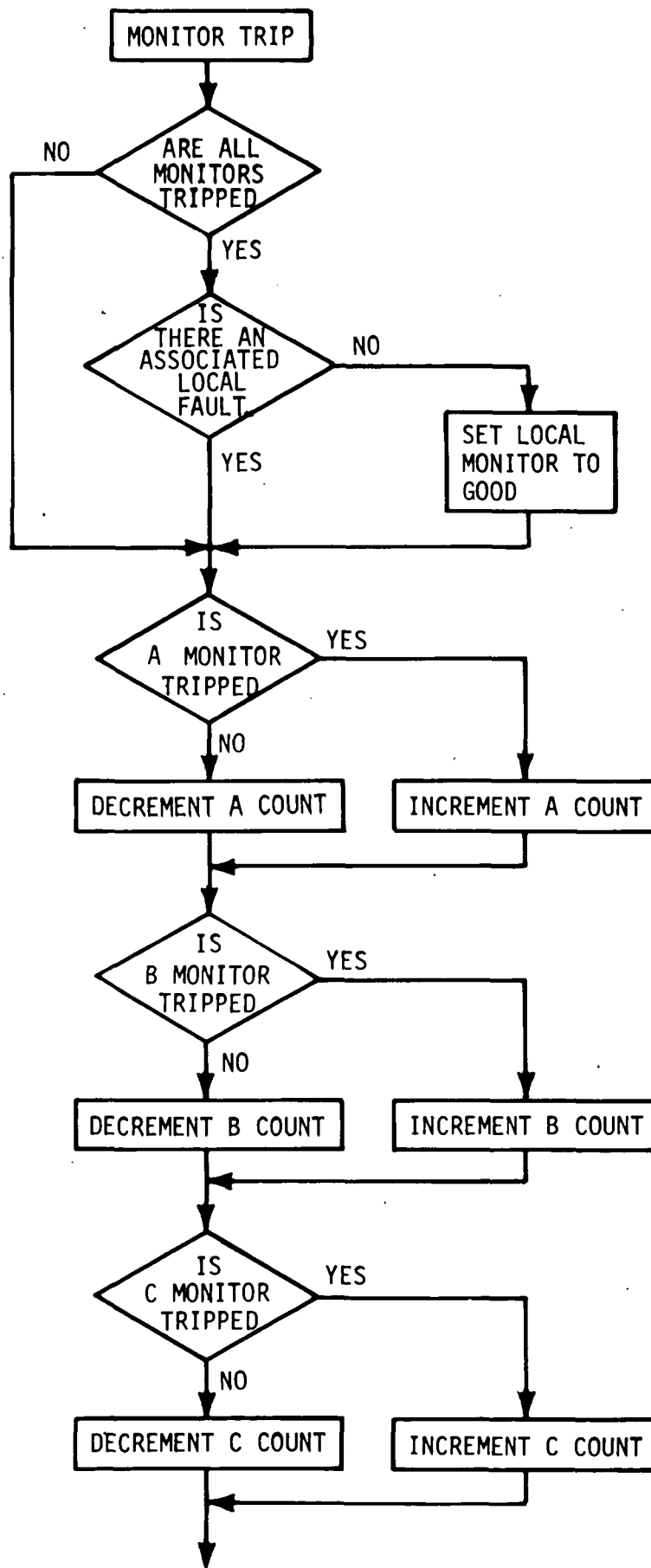


FIGURE B10. TRIPLEX MONITOR TRIP ASSESSMENT

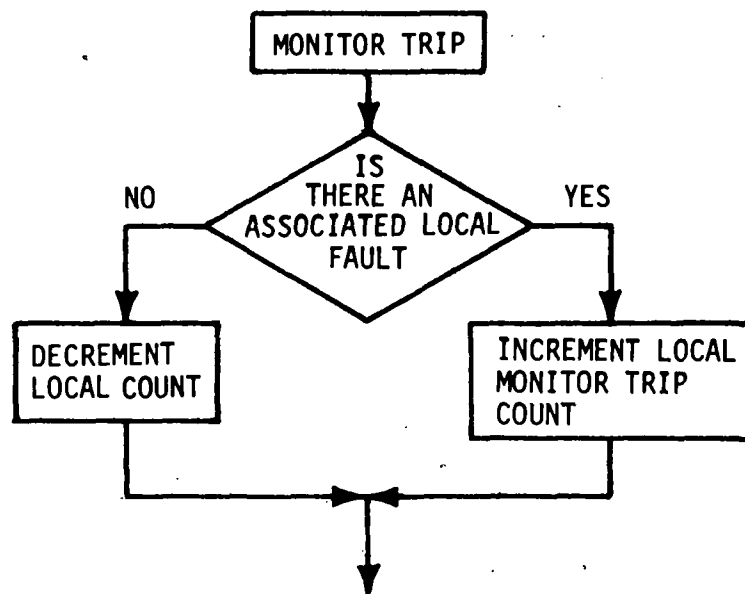


FIGURE B11 DUPLEX MONITOR TRIP ASSESSMENT

Module: ARCS. RELTIM. FORGND. REDMGT. RECFIG. COMPUT. FALSTA;
Failure Status Assessment

- a) Description: The failure status assessment consists of comparing the monitor trip counts to the count thresholds given in Table B6. In the case of the computed servo commands setting a local permanent failure this will result in disengaging the affected servo and degrading that monitor to the next lowest redundancy level. In the case of a foreign servo output monitor trip the associated monitor will be degraded. For the local minor frame or flight mode permanent fault the watchdog monitor will be tripped after a permanent fault is declared.

Figure B12 shows the operation of the failure status assessment. The permanent flags that are set are defined in Table B7.

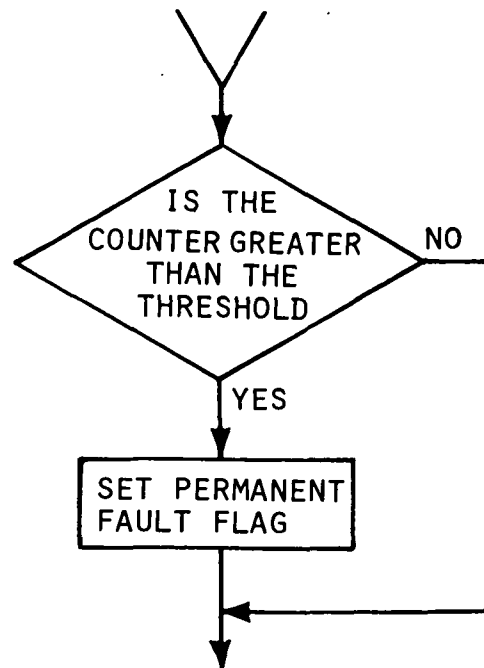


FIGURE B12 GENERAL FAILURE STATUS ASSESSMENT

Permanent Fault Flags	Failure Declared
PMPCS	Pitch Control Surface
PMUR	Upper Rudder
PMLR	Lower Rudder
PMLA	Left Aileron
PMRA	Right Aileron
PMLMS	Left Midspan Spoiler
PMRMS	Right Midspan Spoiler
PMLTS	Left Tip Spoiler
PMRTS	Right Tip Spoiler
PMLWTFC	Left Wing Tip Flutter Control
PMRWTFC	Right Wing Tip Flutter Control
PMLOFS	Left Outboard Flutter Suppressor
PMROFS	Right Outboard Flutter Suppressor
PMFRAME	Minor Frame Count
PMMODE	Flight Mode

} Far
Term

Table B7: Permanent Failure Flags

Module: ARCS. RELTIM. FORGND. REDMGT. RECFIG. XCHMON;

Cross-Channel Monitor

- a) Description: The cross-channel monitor process consists of comparing the local channel integrity word to those received from the foreign channels. In the event that the system is simplex the cross-channel monitor does nothing. The output of this process is a word that indicates the validity of the cross-channel data in the left and right receivers.

Module: ARCS. RELTIM. FORGND. REDMGT. RECFIG. SERVO: Servo Monitoring and Control

- a) Description: A functional breakdown of the servo management task is shown in Figure B13. The function of status assessment is comprised determining the failure and engagement status of the servos. The hardware servo monitor function takes care of the simplex and duplex cases. In triplex there is a requirement for software servo monitoring of the servo sum current (Δp) to access servo operation. The following paragraphs discuss the servo functions shown in Figure B13.

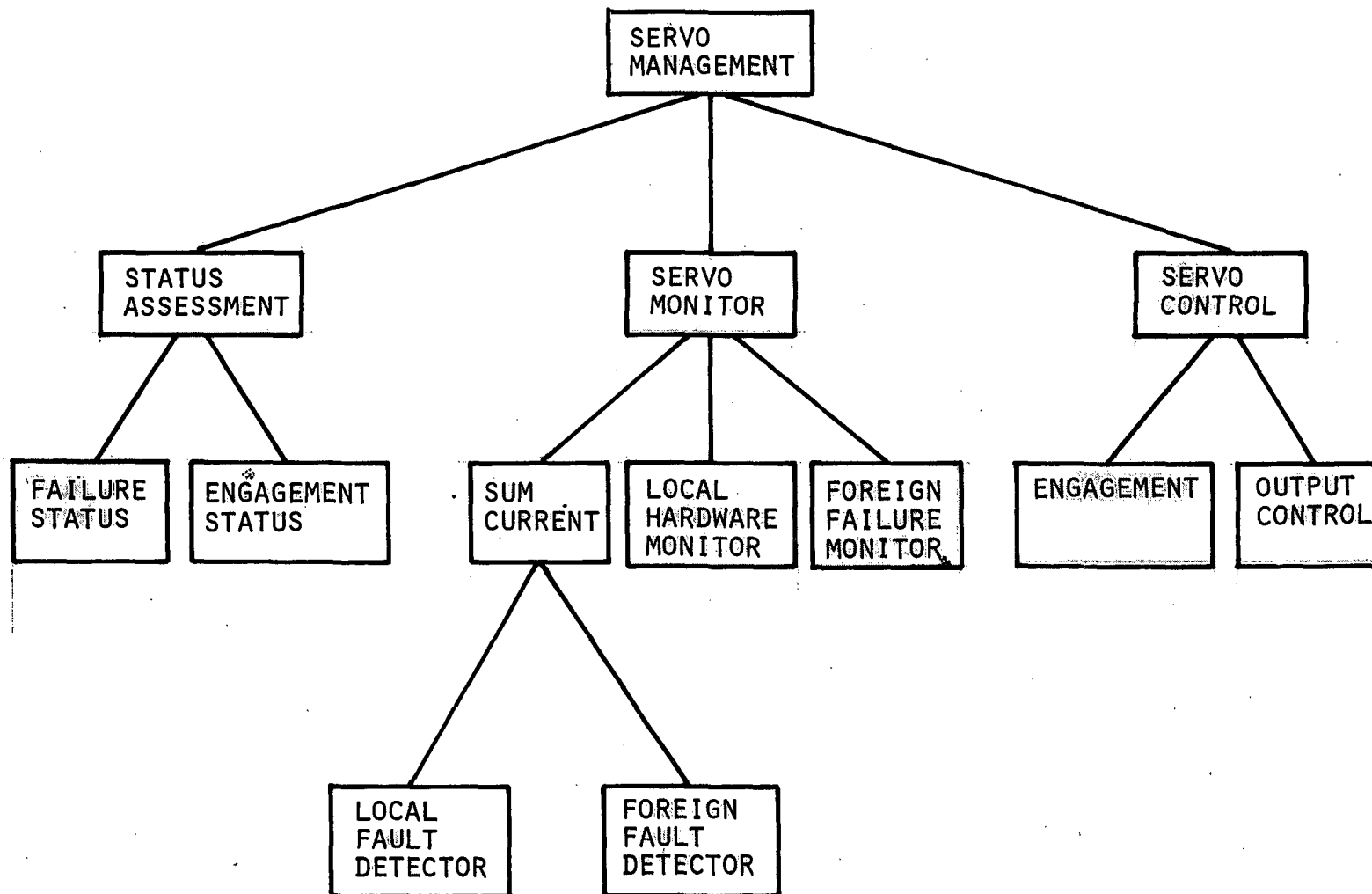


FIGURE B13 SERVO MANAGEMENT FUNCTIONAL BREAKDOWN

Failure Status Assessment — This function is a test of the current failure status of the servo system. If any failures have been registered the system is no longer in a triplex configuration and hence no attempt will be made to software monitor servo operation. Under these conditions control will be transferred directly to output control, otherwise control will proceed through sum current monitoring.

Engagement Status Assessment — If no failures have been registered, yet one or more of the foreign channels have been disengaged, a vote of the three sum currents will not be made and the affected servo fault condition will be registered by incrementing its associated fault counter.

Foreign Sum Current Fault Detector — The three actuator sum currents (proportional to Δp) will be brought together and voted for the purpose of localizing a faulted actuator. If a foreign channel actuator is detected as having faulted, control will be transferred to the Foreign Failure Monitor where a fault record will be made in the form of a fault counter.

Local Sum Current Fault Detector — Local servo faults detected by the sum current comparison will be accumulated by a local fault counter which is a part of the Engagement Control. If no fault is detected control will transfer to a check of the local hardware failure monitors.

Foreign Failure Monitor — Fault status information from the foreign channels will be gathered by the foreign failure monitor from the engagement checks and the sum current monitor. Fault status will be accumulated in the form of a fault counter. When the fault count exceeds a pre-determined maximum limit a foreign failure will be declared, and the servo management function will then, on the next pass, revert to duplex mode of operation which means no software servo monitoring.

Local Hardware Monitor Checks — If the sum current comparison indicates a "no fault" condition, control will transition to a check of the local channel's hardware monitors. The engagement status in the local channel is checked. If it is engaged the local fault counter will be decremented. (The fault count will be limited to a minimum value of zero) If the actuator is disengaged with a good sum current comparison, further checks will be made of the watchdog monitor, power supply monitor, and software permanent fault flags status. If the disengagement was caused by either of the hardware monitors, there is nothing that the servo management can do to affect servo engagement. However, if both of these monitors are indicating a good state an attempt will be made to re-engage the servo actuator if the software flag is reset by commanding a shutdown followed by re-engagement. The result of this action is that a reset pulse will be sent to the servo, which, if the condition which caused the disengagement has been cleared, will cause the servo actuator to be re-engaged.

Engagement Control — This portion of the servo management function has control over the shutdown and/or engagement of the servo actuator. A permanent shutdown will result either when the fault counter exceeds its maximum allowable level or a computer failure has been declared by the output monitor thereby dictating that the associated servo be disengaged. Servo re-engagement will be allowed only following a transient fault of the independent hardware monitor or a temporary loss of hydraulic pressure.

Output Control — This process simply consists of a test of the computer fault status from which the decision is made whether or not to update the servo command output. For example, in duplex operation, when an output fault is detected by the output monitor, both systems outputs can be held until a fault location decision can be made.

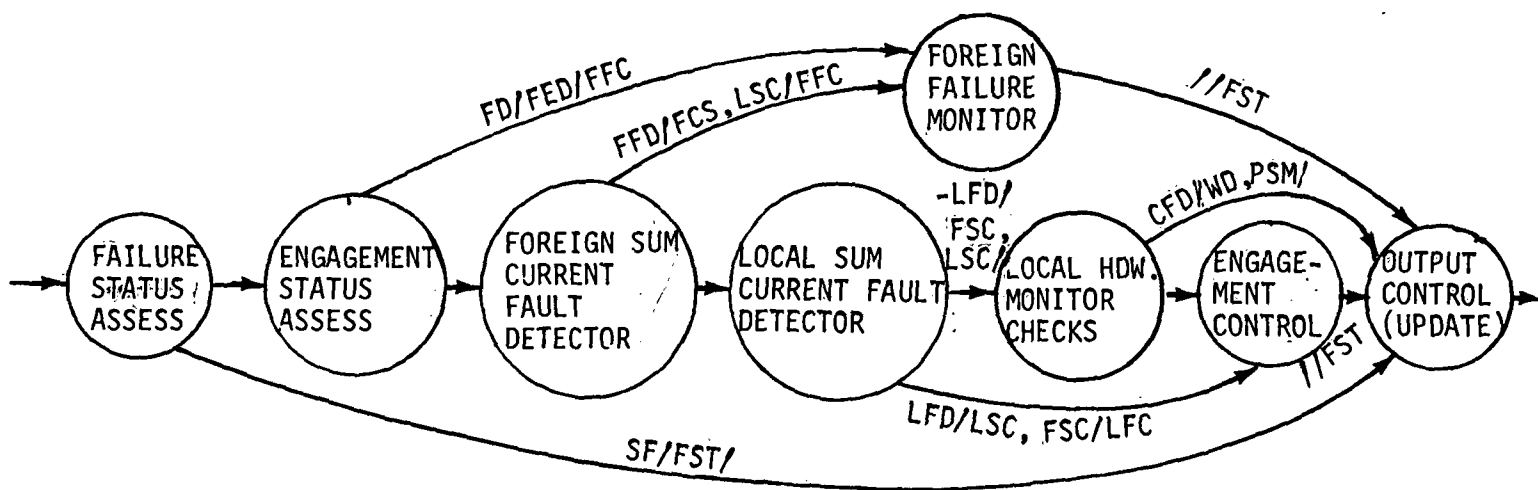
b) None

c)

d) Data-Element Definitions:

FED	—	Foreign Engagement Discretes
FSC	—	Foreign Sum Currents
LSC	—	Local Sum Currents
FST	—	Failure Status Table
FFC	—	Foreign Fault Counter
LFC	—	Local Fault Counter
WD	—	Watchdog Monitor Flag
PSM	—	Power Supply Monitor Status Flag

e) Transition Diagram:



f) Conditions:

SF	— Servo Failure
FD	— Foreign Channel Disengaged
LFD	— Local Fault Detected
FFD	— Foreign Fault Detected
CFD	— Computer Fault Detected

B.1.3 Application Software Modules

The ARCS application software modules perform the processes of control law computation and mode control logic for those control laws.

Module: ARCS. RELTIM. FORGND. MODCON; Mode Control

a) Description: The mode control tasks are mode select, mode logic and mode annunciation. Mode control inputs are discrete inputs representing pilot selections, airplane discretes and control law discretes. Outputs are flags that determine the set of control law algorithms to be processed as a result of the pilot's selections, the airplane configuration, flight condition, and progress along the flight path.

b) Sub-processes:

MODSEL — Mode selection

MODLOG — Mode logic

MODANN — Mode annunciation

c) Inputs and Outputs of the Sub-processes:

MODSEL	
Inputs	Outputs
MODIN	SELOUT

MODLOG	
Inputs	Outputs
SELOUT	MODE
LAWDIS	ANNIN

MODANN	
Inputs	Outputs
ANNIN	ANLITE

d) Data-Element Definitions:

(MODIN — FORGND)

(MODE — FORGND)

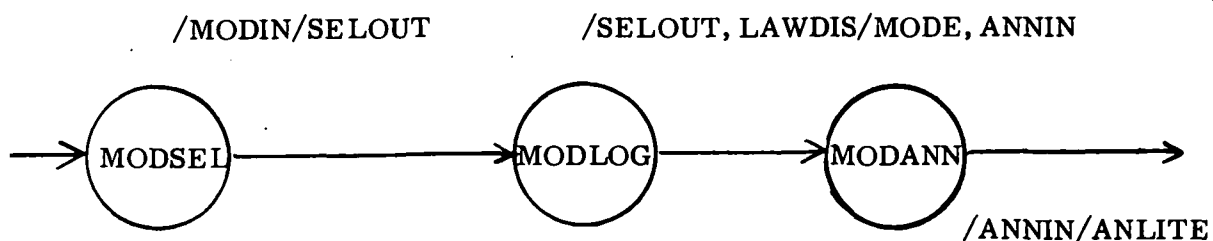
(LAWDIS — FORGND)

SELOUT — Mode selection outputs

ANNIN — Mode annunciation inputs

ANLITE — Mode annunciation outputs

e) Transition Diagram



- a) Description: The near term application model control laws are those selected for the baseline software. The control laws are functionally broken out by the surfaces they control. The control law block diagrams are discussed in Appendix I. Figure B14 shows the functional tree for the control laws. Figure B15 shows the inputs and outputs of the control laws.

The scheduling of the control is a function of the flight mode requested by the pilot and the current minor frame.

- b) Sub-processes:

ELVATR — Elevator Function
 AILRON — Aileron Function
 INBSPL — Inboard Spoilers
 RUDDER — Rudder Function

- c) Inputs and Outputs of the Sub-processes:

ELVATR		AILRON	
Inputs	Outputs	Inputs	Outputs
q	ELEV	Q-pot	
a_z		δ	
$F_{col\ c}$		p	LFTAIL
$F_{col\ F}$		$F_{wheel\ c}$	RGTAIL
h_{baro}		$F_{wheel\ F}$	
V_{GS}		TKA	
δ		HFC	
VPC		RAD_{alt}	
FLAP		LOC	
α		\dot{y}	
GSE		\ddot{y}	
RAD_{alt}		y	
		$\Delta\psi$	
		$\Delta\psi_c$	

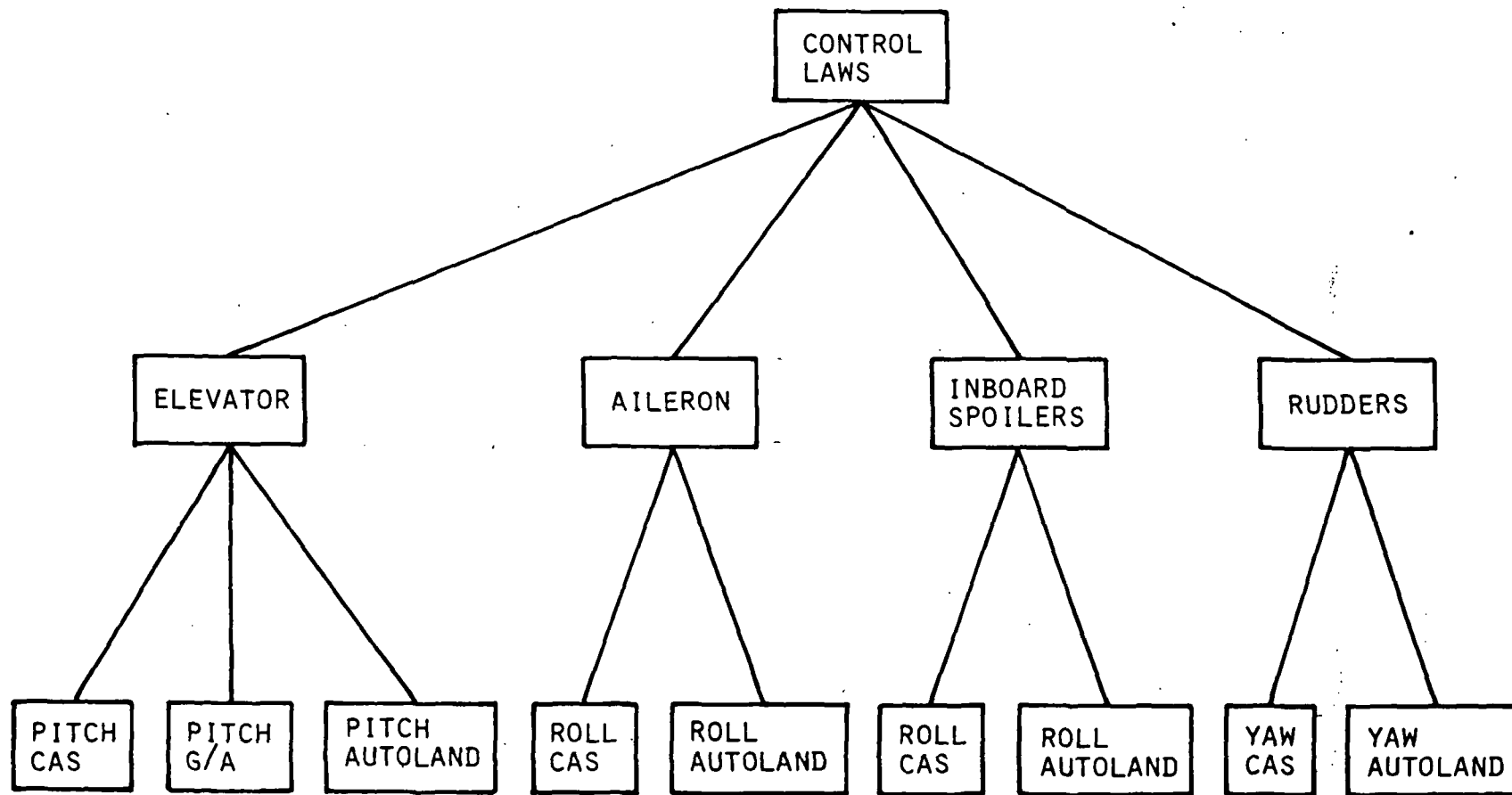


FIGURE B14 CONTROL LAW TREE

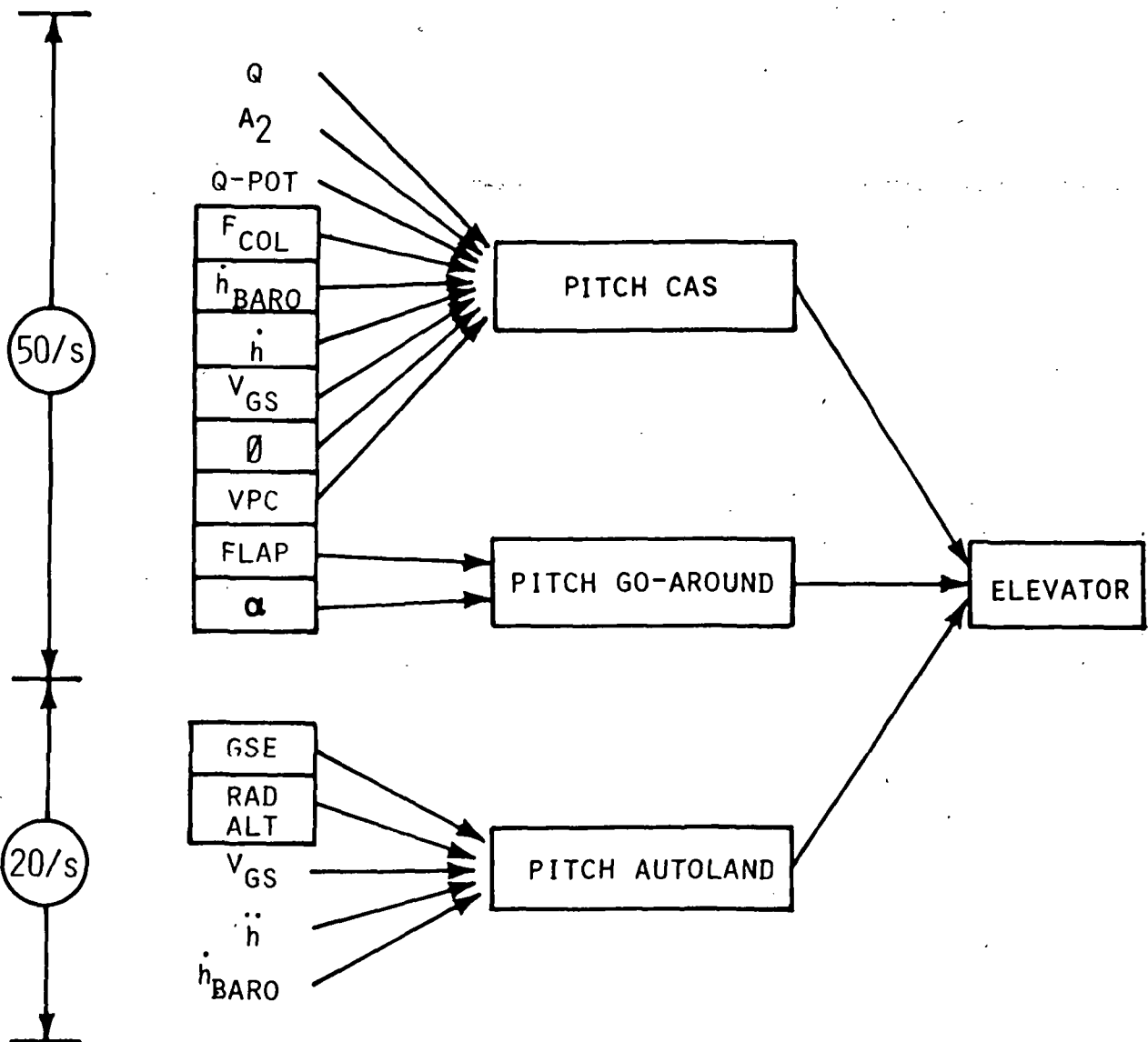


FIGURE B15 INPUTS AND OUTPUTS OF CONTROL LAWS

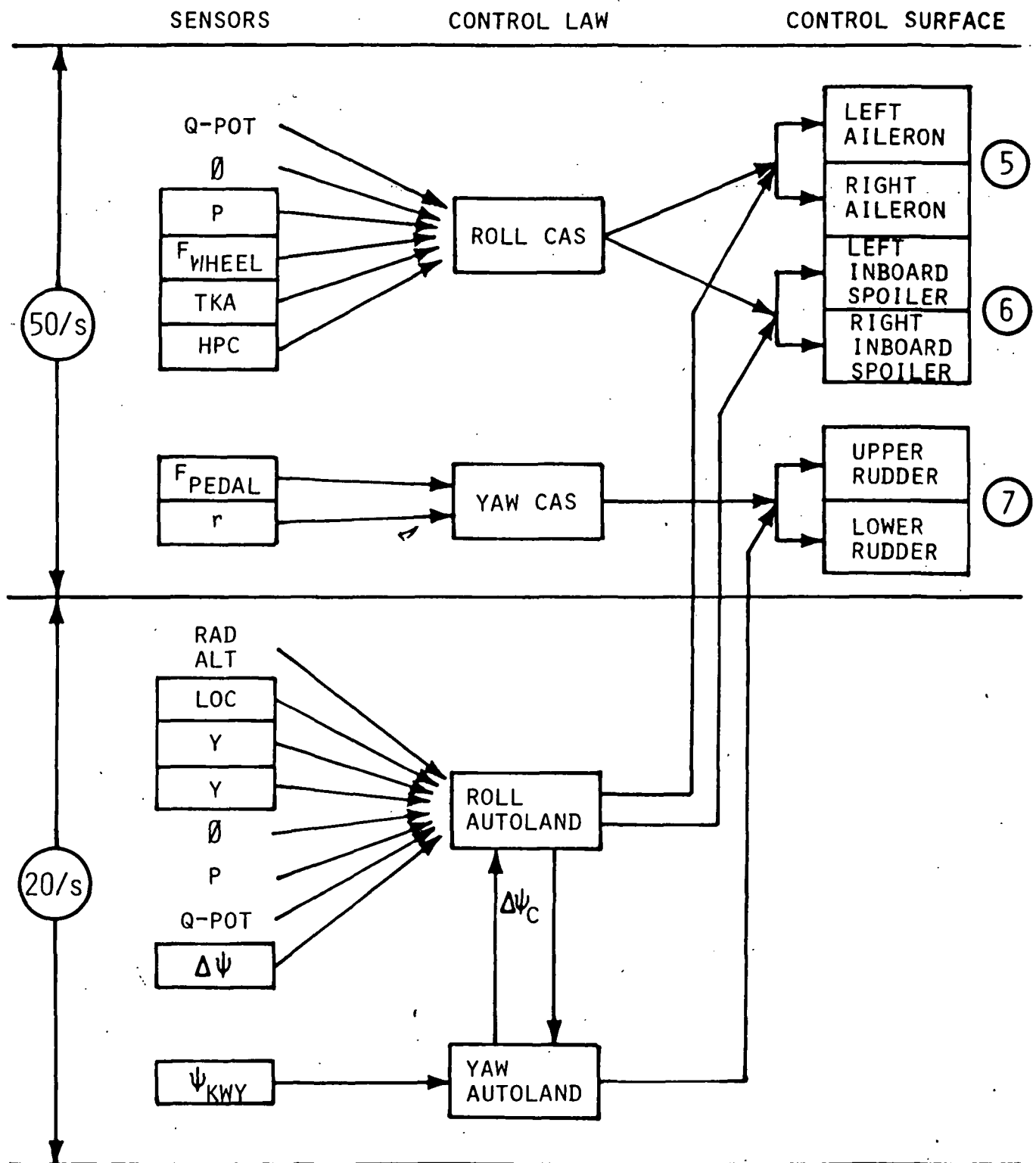


FIGURE B15 INPUTS AND OUTPUTS OF CONTROL LAWS

c) (cont'd)

INBSPO		RUDDER	
Inputs	Outputs	Inputs	Outputs
Q-pot		F _{pedal c}	
ϕ		F _{pedal F}	UPRUD
p		γ	LORUD
F _{wheel c}		ψ runway	
F _{wheel F}			
TKA			
HPC	LINSPO		
RAD _{alt}	RINSPO		
LOC			
\dot{y}			
\ddot{y}			
$\Delta\psi$			
$\Delta\psi_c$			

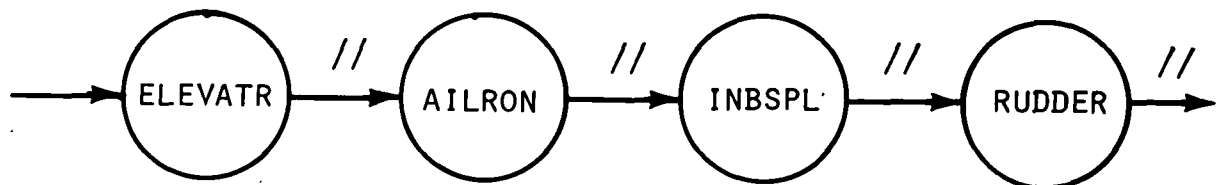
d) The Data-Elements:

LFTAIL — Left aileron command
 RGTAIL — Right aileron command
 ELEVC — Elevator command
 LINSPO — Left inboard spoiler
 RINSPO — Right inboard spoiler
 UPRUD — Upper Rudder
 LORUD — Lower Rudder
 q pitch rate
 a_z normal acceleration (aircraft)
 Q-pot dynamic reference pressure computed from air data
 F_{col c} Captain's column force
 F_{col F} F/O
 h_{baro} barometric altitude rate
 \ddot{h} vertical acceleration (aircraft)
 V_{GS} ground speed
 ϕ bank angle
 VPC vertical path command (from nav computer)
 FLAP flap position information
 α angle-of-attack
 GSE glide slope deviation
 RAD_{alt} radar altitude
 p roll rate

d) (cont'd)

$F_{\text{wheel } c}$	Captain's wheel force
$F_{\text{wheel } F}$	F/O wheel force
TKA	track angle
HPC	horizontal path command (from nav computer)
$F_{\text{pedal } c}$	Captain's pedal force
$F_{\text{pedal } F}$	P/O pedal force
r	yaw rate
Loc	localizer deviation
\dot{y}	cross track velocity
\ddot{y}	cross track acceleration
$\Delta\psi_{\text{rwy}}$	runway heading
$\Delta\psi$	heading

e) Transition Diagram:



B.1.4 System Test

The purpose of system test is to provide preflight tests and in-flight tests. The preflight (or ground tests) are discussed in section B.1.4.1. The in-flight tests are discussed in section B.1.4.2.

B.1.4.1 Ground Test

The ground tests provide preflight monitoring to assess flight worthiness.

a) Description: The ground test operation is composed of interrupt processing and preflight/maintenance testing. Entry into this module will only occur when such a request has been generated by the operator and the aircraft is stationary on the ground. The transition out of real time operation occurs from the background (asynchronous) tasks where both of these criteria are assessed. Ground testing will test and hence ascertain the operational integrity of the entire ARCS system including sensors, computers and servos.

b) Sub-Processes:

PREMAN	— Preflight/maintenance testing
INTPRO	— Interrupt processing

- a) Description: Though ground test is primarily an asynchronous operation the timer interrupt normally serving as the synchronization time base is still allowed to occur to serve as a time base for those tests which are time critical. Also upon the occurrence of a timer interrupt a test is made of the current "on-ground" status and the STP request buffer. The test program is exited if the on-ground test is false or if the test operator has requested that the test be ended. Following the processing of the timer interrupt control will return either to a continuation of ground test or to real time control as shown in Figure B16.

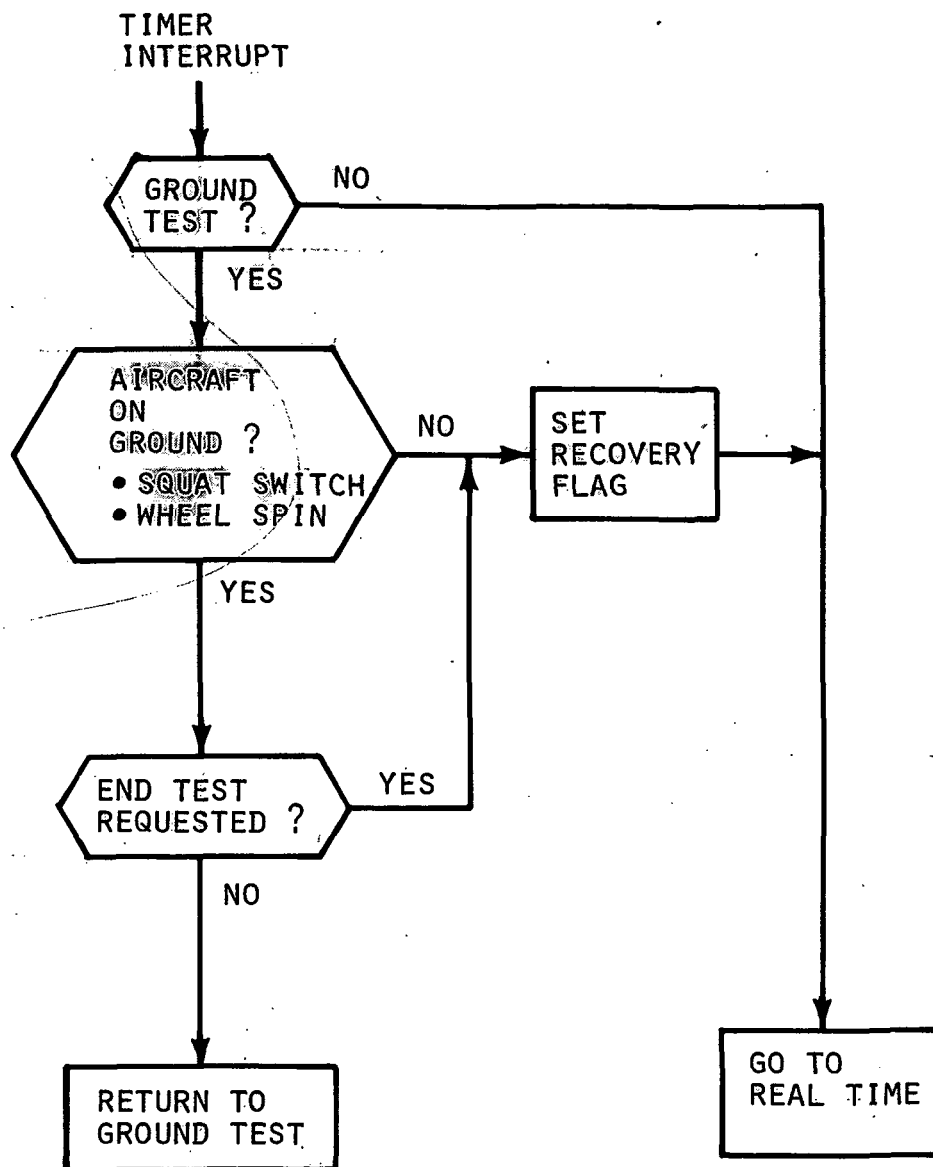


FIGURE B16 GROUND TEST INTERRUPT PROCESSING

- a) Description: Ground tests are comprised of both the preflight test and post flight or maintenance test functions. The functional tree for this module is shown in Figure B17.

The functions which will be included in the preflight test can be categorized into two groups: those that are dispatch required as defined by the airframe manufacturer and Federal Aviation Administration due to flight safety considerations; and those mandated by the individual airline as needed for their specific operations. Maintenance level testing on the other hand must include all aspects of the AFCS including non-critical sensor and servo systems.

A request for ground test is generated by the operator via the System Test Panel (STP). This information is acknowledged by the Background Test via the STP input data table. On-ground status is determined by examination of two discrete status signals, the main landing gear squat switches and the wheel spin discrete. Activation of the ground test function is inhibited unless all discretes show the aircraft to be stationary. Following completion of ground test, control will be returned to Real Time Operations. Ground testing will be terminated whenever either the test procedure is completed in a normal manner or the ground test function is deselected by the test operator or the on-ground status ceases, as would be the case if the aircraft begins to roll. This is effected by setting the "Recovery Flag" prior to exiting from the ground test program which will force the Real Time Operation to perform a RAM initialization followed by recovery synchronization.

- b) Sub-Processes:

INITIAL	—	Initialize
TEST	—	Test
DISPL	—	Display

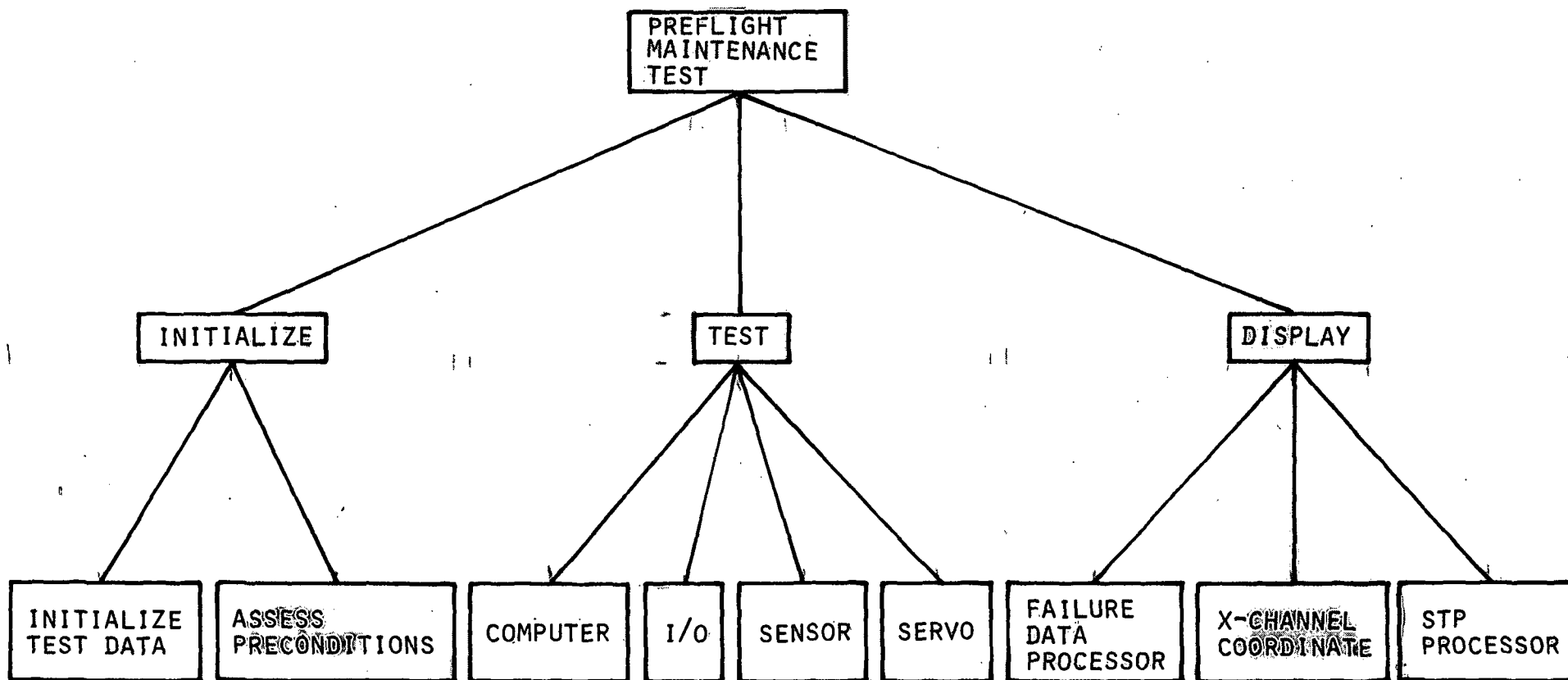
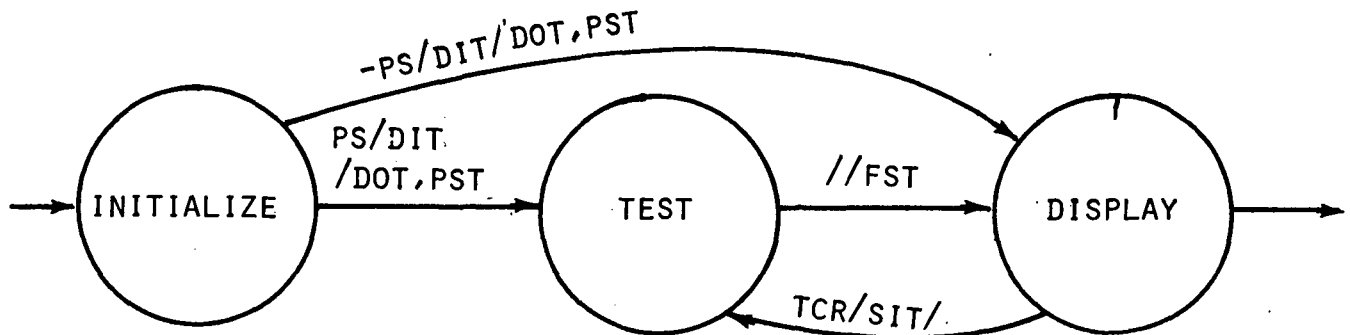


FIGURE B17 MAINTENANCE TEST FUNCTIONAL REQUIREMENTS

d) Data Elements:

SIT — STP Input Table
DOT — Discrete Output Table
PST — Precondition Status Table
FST — Failure Status Table
DIT — Discrete Input Table

e) Transition Diagram:



f) Conditions:

PS — Preconditions Satisfied
TCR — Test Continue Request

Module: ARCS.GNDTST.PREMAN.INITAL: Initialize

- a) Description: Initialization consists of all those tasks necessary to put the system into a configuration from which testing can be safely and accurately conducted. This involves both the conditioning of RAM data and the assessment of required preconditions. Conditioning of RAM data includes forcing a bypass of all servo actuators since during the conduct of the test all aspects of the computer subsystem will be exercised in some fashion and to allow servo actuators to be engaged during this time presents a potentially hazardous condition. For ARCS System Test the assessment of preconditions reduces to two tasks: to check the interface to and from the STP, and to check for the existence of sensor valid discretes.

If the interface link to or from the STP is inoperative or the STP itself has failed no further testing will be attempted. Following confirmation of the operational integrity of the STP and its associated data links each of the sensors are then checked for the existence of a proper validity indication. It is assumed for this check that the absence of a valid signal at this point in the test is most likely due to lack of power to the associated sensor. This situation will then be brought to the attention of the test operator via the STP who can then take corrective action.

Module: ARCS.GNDTST.PREMAN.TEST: Test

- a) Description: The test function (see Figure B18) consists of computer tests; input and output electronics tests, sensor tests and servo tests. Entry into this module comes from initialization and exit can occur either from a "de-selection" of the ground test function or upon its completion. At the completion of the test function sufficient information is available to ascertain the operational status of the total system.

The procedure of ground testing makes use of a so-called "center out" test philosophy wherein the most basic elements are tested first and then these basic elements are used to test other functions. At the highest level, this test structuring results in an organization which first tests the computer LRU followed by sensors and concluding with servo systems. The computer testing segment can be further broken down into its constituent parts, i.e., central processor element, RAM and ROM memories, and input/output elements.

b) Sub-Processes:

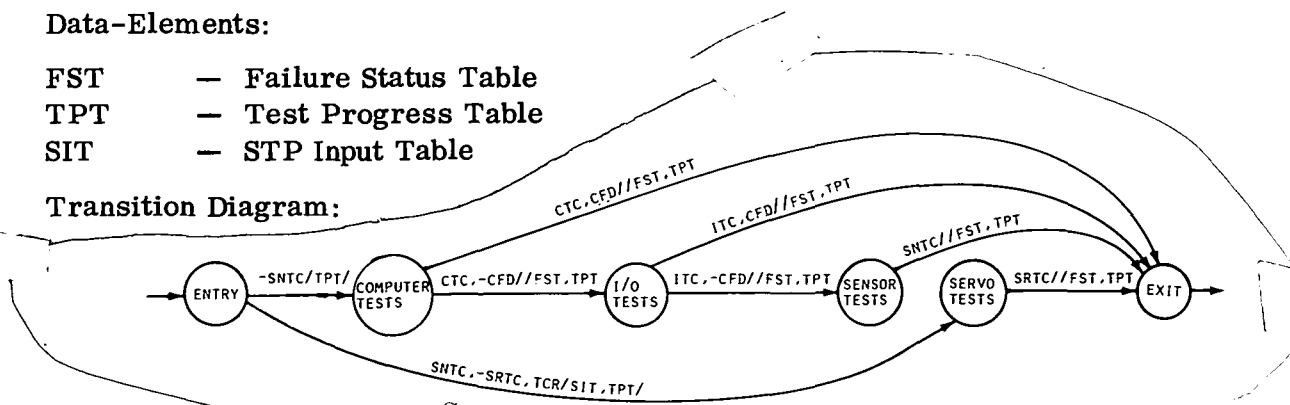
COMTST	— Computer Test
IOTST	— Input/Output Test
SENSOR	— Sensor Test
SERVO	— Servo Test

c)

d) Data-Elements:

FST	— Failure Status Table
TPT	— Test Progress Table
SIT	— STP Input Table

e) Transition Diagram:



f)

Conditions:

CFD	— Critical Failure Detected	SNTC	— Sensor Tests Complete
CTC	— Computer Tests Complete	SRTC	— Servo Tests Complete
ITC	— I/O Tests Complete	TCR	— Test Continue Request

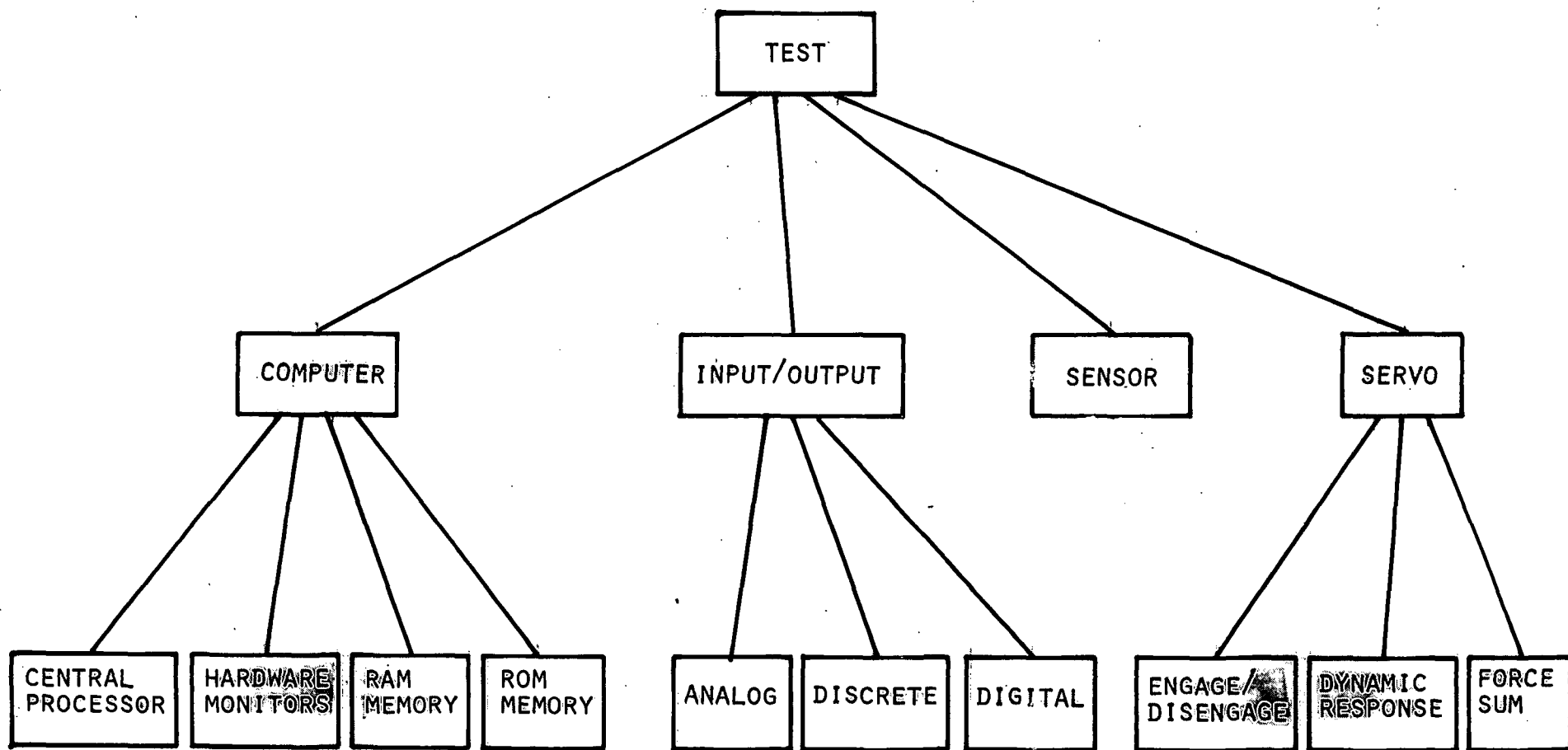


FIGURE B18 TEST STRUCTURE

- a) Description: Computer testing includes tests of the central processor, hardware monitors, ROM memory, and RAM memory. The computer unit, in addition to being the single most critical element in the ARCS control system, serves as the test administrator for the rest of the System Test. Therefore its own integrity must be verified before it can be called on to test peripheral systems. By similar argument, the testing of the computer unit begins with a diagnostic of its central processor unit (CPU).

Processor self-test performs an instruction test sequence which tests all instructions (or at least all micro-instructions), all registers, and all addressing modes.

The ability of each of the hardware monitors to detect and enunciate the existence of a fault condition is next verified to assure that the system is free of latent failure conditions.

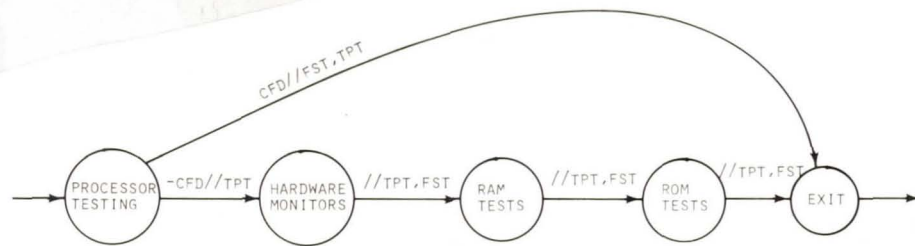
The ROM or constant memory will be tested using sum checking techniques wherein all of the words in the program memory are summed together in one 16-bit word ignoring any overflows that occur. The resultant sum is then compared to the known constant 16-bit sum. RAM memory is checked by writing into and reading out of all accessible RAM memory a predetermined data pattern and monitoring the pattern read out as compared to the pattern which was written in.

c)

- d) Data-Elements:

FST	—	Failure Status Table
TPT	—	Test Progress Table

e) Transition Diagram:



f) Conditions:

CFD — Critical Failure Detected

Module: ARCS.GNDTST.PREMAN.TEST.IOTST: Input/Output Test

- a) Description: Analog, discrete and digital inputs and outputs are tested utilizing what is commonly referred to as wraparound testing whereby a specific output is commanded by the processor and the resultant output looped back into the corresponding input processor. The received data can then be compared to what was transmitted to check both input and output functions.

- a) Description: To verify the operational integrity of the ARCS sensor system the sensor testing process involves stimulating the sensor outputs to produce some expected output value. Wherever possible the sensor's own internal selftest is used to this output by setting the appropriate computer output discrete to cause automatic sensor selftest stimulation. When automatic stimulation is not possible, the needed sensor output is generated by the test operator. This is accomplished by formatting and displaying a cue to the test operator indicating the required action.

For sensor systems, particularly analog equipment, the expected response time to selftest stimulation may require wait times on the order of several seconds. For this reason the sensor testing portion of the ARCS ground test operation is a parallel operation where testing of several systems may be in process simultaneously. The test program for each sensor must therefore be broken down into a series of subtasks each of which can be completed within its allocated portion of the total frame time.

Module: ARCS.GNDTST.PREMAN.TEST.SERVO: Servo Testing

- a) Description: Servo testing (see Figure B19) includes synchronization, engagement control testing, dynamic response tests and force override tests. Upon completion of sensor testing the local test completion status will be communicated cross channel and the test program will move on into the display portion. Servo testing is one test area which necessarily requires a coordinated approach between all operable computers. Testing will not proceed until it is ascertained that all operable systems have completed sensor testing and are now ready to proceed. At this point the display function shall cue the operator that the system is ready to commence with servo testing and request further direction. The System Test function will enter a wait loop while monitoring the STP input table for operator inputs.

The first task to be accomplished in servo testing is to establish a regular time reference to satisfy watchdog monitor requirements. This is effected by each processor initiating a new iteration timer count and establishing a reference time for the watchdog monitor, upon receipt of a test continue request from the STP. Thereafter, whenever an iteration timer interrupt occurs, a synchronization process similar to the normal real time synchronization will occur.

b)

c)

d) Data-Elements:

DIT	— Discrete Input Table
DOT	— Discrete Output Table
FST	— Failure Status Table
TPT	— Test Progress Table

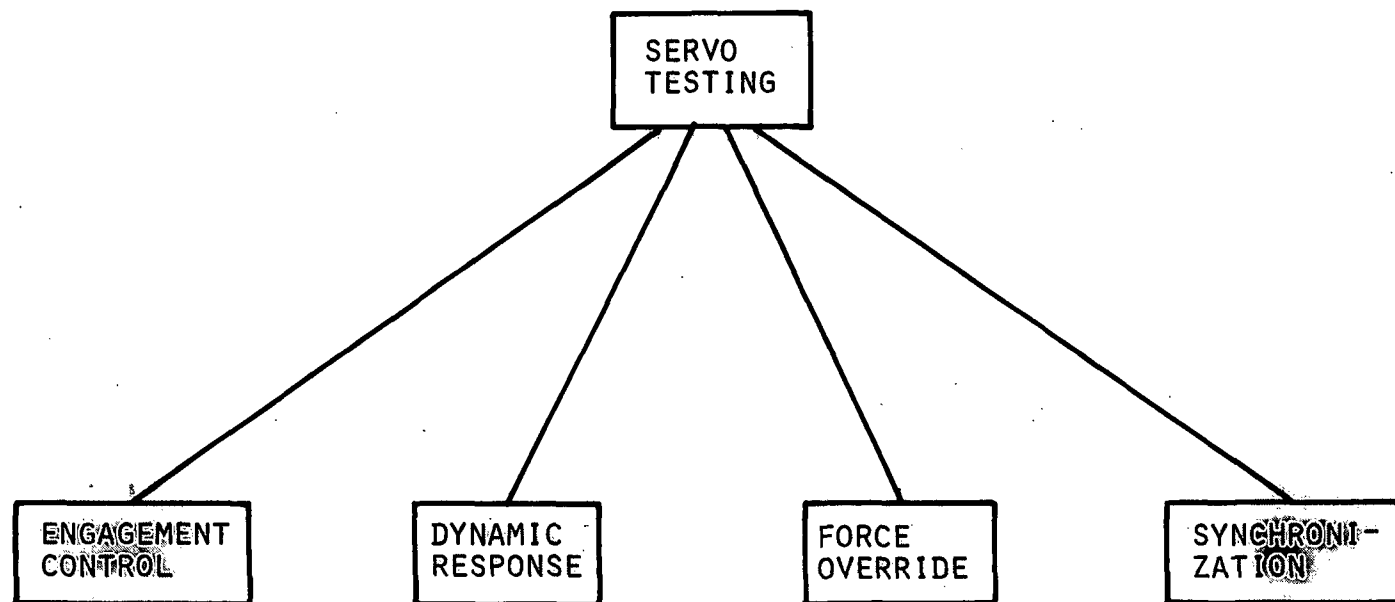
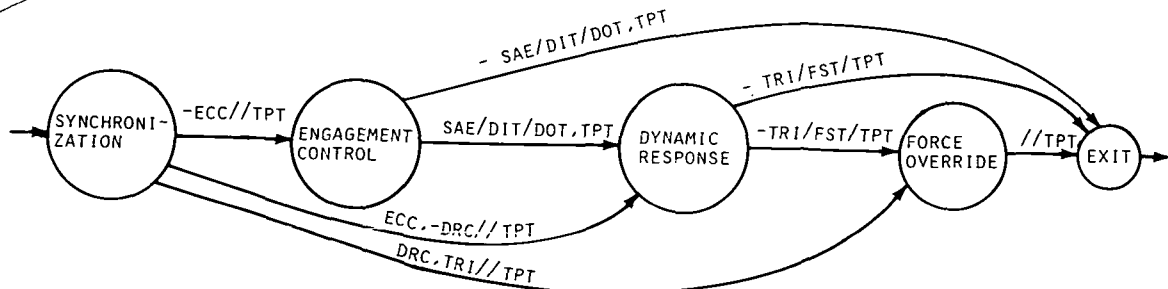


FIGURE B19 SERVO TEST FUNCTIONAL BREAKDOWN

e) Transition Diagram:



f) Conditions:

SAE	—	Servo Actuator Engaged
TRI	—	Triplex Configuration
ECC	—	Engagement Control Complete
DRC	—	Dynamic Response Complete

Module: ARCS.GNDTST.PREMAN.DISPLY: Display

- a) Description: The display function (see Figures B20) and B21) is comprised of three processes, failure data processing, cross-channel communication, and STP processing. Its purposes are to compile failure data generated by the Initialization and Testing functions, present them to the operator in a meaningful useful manner, accept request inputs received from the operator via the STP, and direct the testing sequence accordingly.

Failure processing consists of the accumulation and formulation of test progress and failure data from the various tables used by the ground test function, and deducing the information to be communicated outside. Upon entry into the failure processing state an assessment must be made as to what was the condition detected that initiated the control transfer to determine the appropriate response. The response includes formatting the information for cross-channel transmission and /or display on the STP.

The state identified as Cross Channel Results is the vehicle by which the test results, as derived by the local computer, are communicated to all other operable computers. The state identified as STP processor performs the two tasks required to interface the System Test function with the test operator - format and transmit display data and process incoming operator commands.

ORIGINAL PAGE IS
OF POOR QUALITY

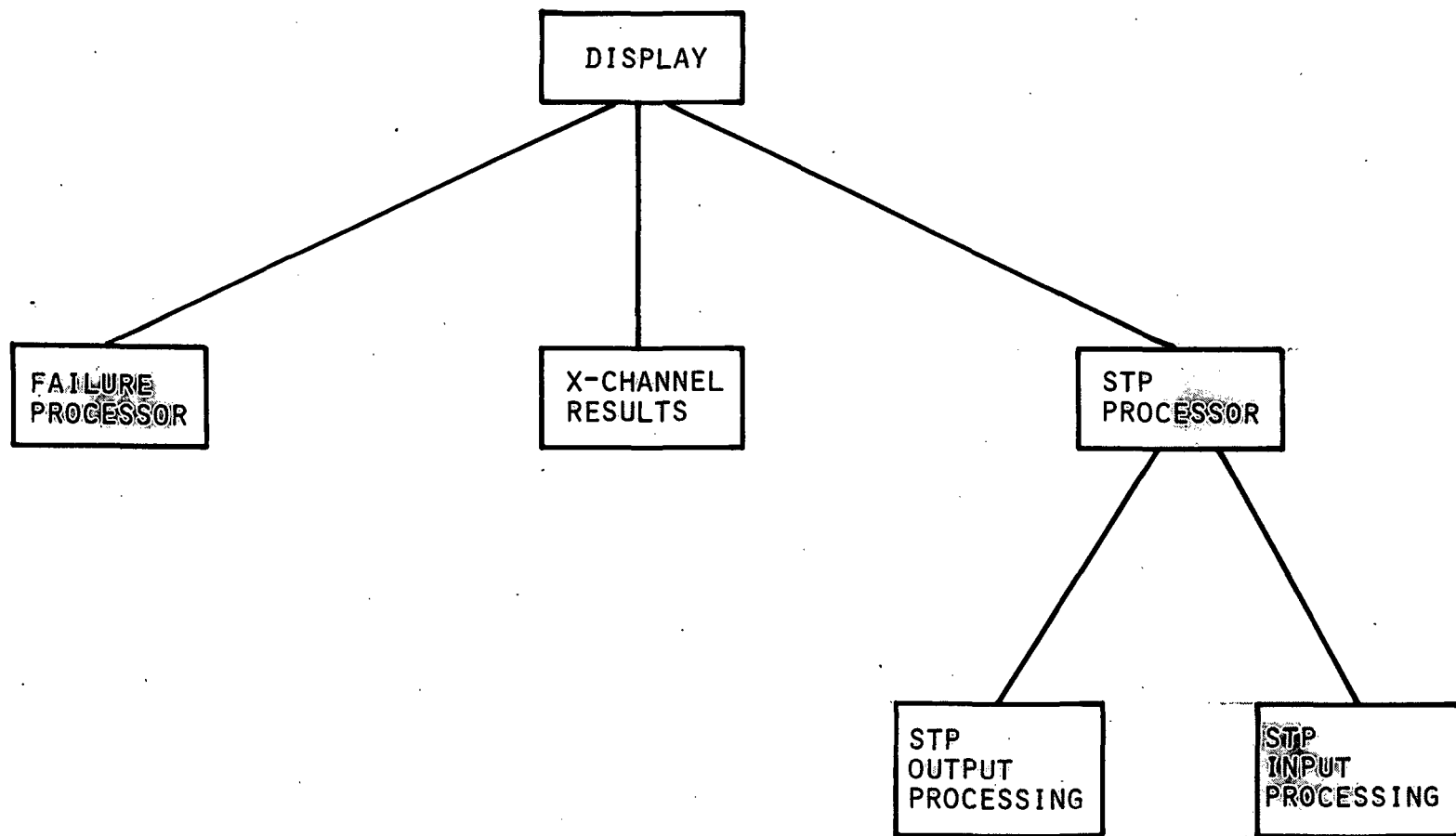


FIGURE B20 DISPLAY FUNCTION

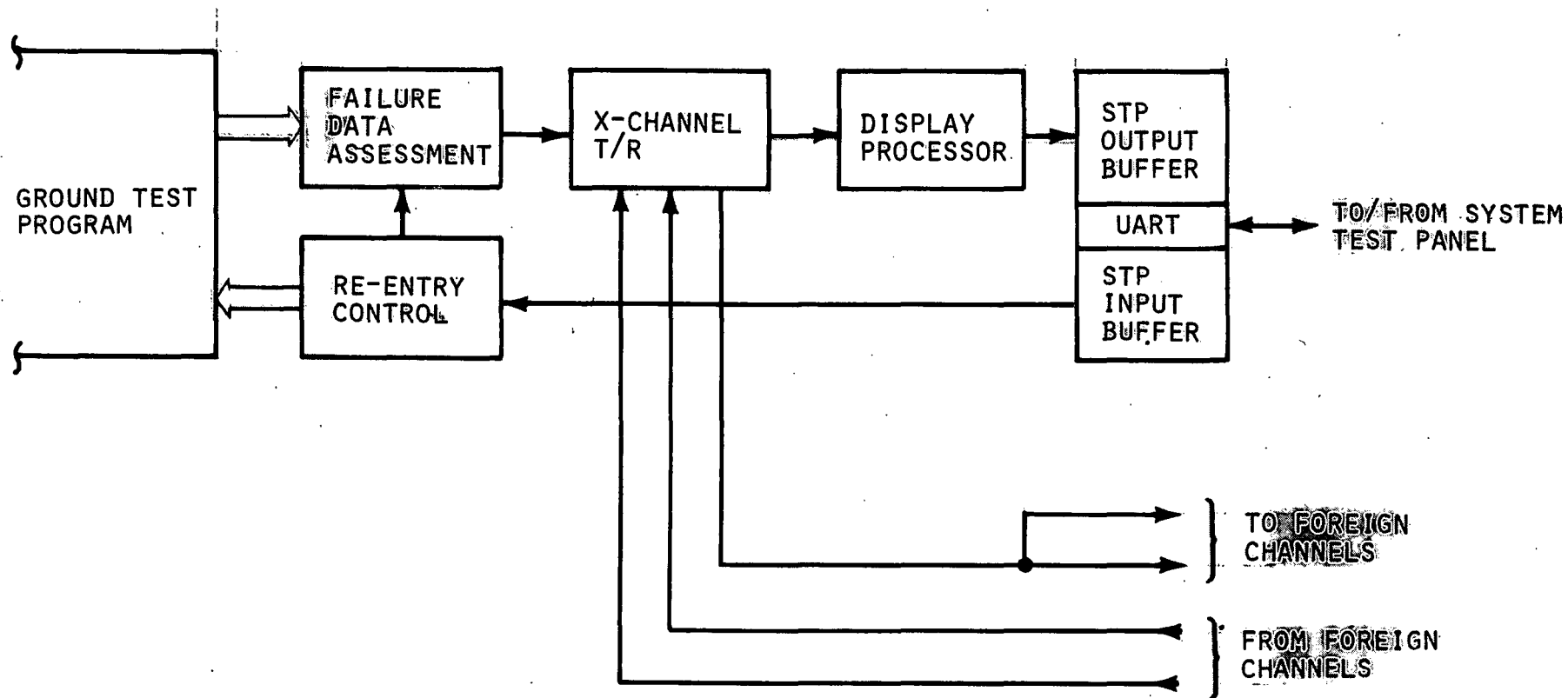


FIGURE B21 GROUND TEST DISPLAY OPERATION

B.1.4.2 Background Tests

The background tests provide in-flight testing which is used to help localize faults and to provide maintenance information.

Module: ARCS.RELTIM.BAKGND: Background Tasks

- a) Description: The background tasks consist of background entry control, on-line self-test, maintenance data update, and system test panel (STP) processing. During any particular frame, the first operations to be executed will be the foreground tasks required for that frame. When the foreground tasks are completed the remaining time in the frame will be used to execute background tasks.

In-flight (or on-line) testing of the computer subsystem is the primary function performed by the Background Task. Its purpose is to detect and/or localize those failure conditions which might not be detected by first level system monitors. The in-flight test is a continuous, repetitive check of the computer subsystem which keeps a running record of anomalies within the local computer.

Background entry control serves as the task scheduler and directs control to the particular function to be executed next. Maintenance data update gathers all system failure data for maintenance purposes. STP processing serves as the interfacing link outputting display data and accepting operator requests.

- b) Sub-Processes:

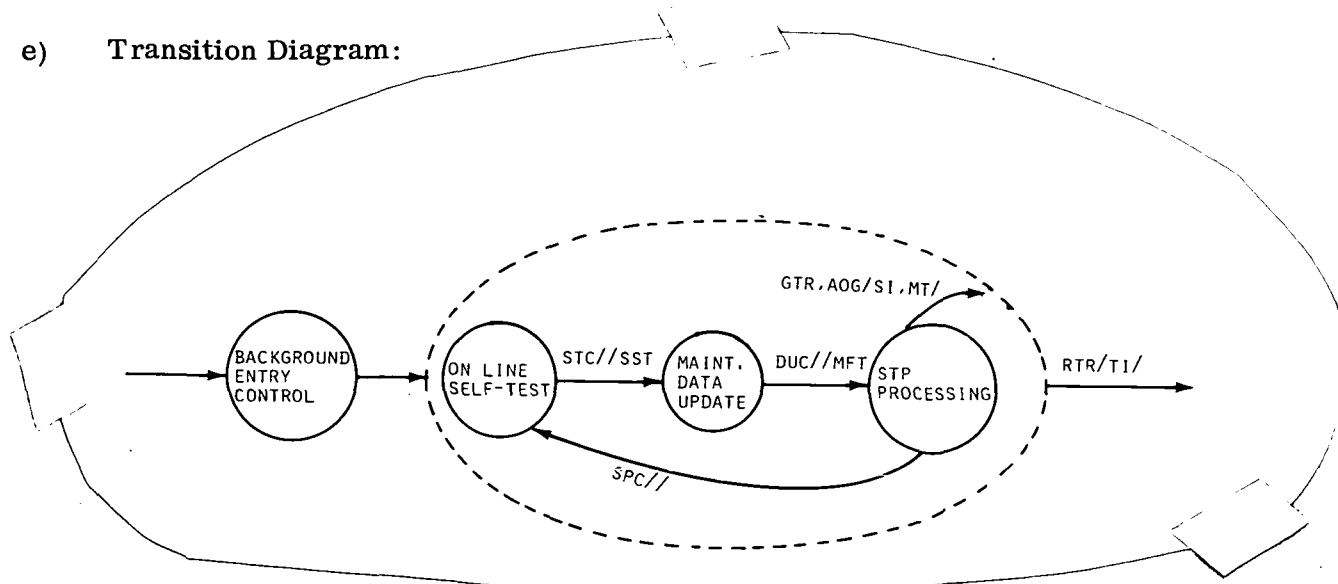
ENTCONT	— Background entry control
ONLINE	— On-line self-test
MANDAT	— Maintenance data update
STPPO	— STP processing

- c)

- d) Data-Elements:

SI	— STP Input Table
MT	— Mode Table
TI	— Timer Interrupt
SST	— System Status Table
MFT	— Maintenance Failure Table

e) Transition Diagram:



f) Conditions:

- | | | |
|-----|---|-------------------------|
| GTR | — | Ground Test Requested |
| AOG | — | Aircraft on Ground |
| RTR | — | Real Time Required |
| STC | — | Self-Test Complete |
| DUC | — | Data Update Complete |
| SPC | — | STP Processing Complete |

Module: ARCS.RELTIM.BAKGND.ENTCON: Background Entry Control

- a) Description: Background entry control acts as the task scheduler and directs control to the particular function to be executed next, based on the last instruction executed in the previous frame. Depending on the nature of the particular test being executed, it may be possible to simply start up on the next instruction in sequence or it may be necessary to backtrack to the start of the test and begin again on the next frame. The entry state in addition to scheduling the background testing sequence must also include a deadline timer to provide assurance that the background tasks are being executed on a regular basis. The deadline timer consists of a frame counter which gets incremented every time the background tasks are entered. The counter must be periodically reset by the background test program or else a counter overflow will occur and a fault condition indicated.

- a) Description: The three processes involved in on-line self-testing are: (1) processor self-test/diagnostics, (2) memory testing, and (3) input and output testing. Processor self-test is the primary software self-test for the computer unit. Although it runs in the background mode, it must be completed repetitively as determined by the time deadline, i. e., a maximum time is specified between sequential completions of the test. The computer self-test begins with an instruction test sequence within which all instructions are exercised, all registers involved, and all addressing modes used. If the processor performed an erroneous computation and goes into one of these endless loops the time deadline for the computer self-test would not be met.

The second test to be performed in background mode is a program memory sum check, wherein all of the words in the program memory are summed and compared to the known correct sum.

The third test in the computer self-test sequence is a scratchpad read-write test. A number of locations in the scratchpad are dedicated to self-testing. On successive iterations of the test, random patterns are written into these dedicated locations and then checked.

Proper operation of the computer input and output sections is tested using wrap-around loop checks of both analog and discrete data.

Module: ARCS.RELTIM.BAKGND.MANDAT: Maintenance Data Update

- a) Description: Maintenance data update functions as a central collection point of all failure data for maintenance purposes. Here all of the failure information accumulated during the previous in-flight test loop is assessed along with the failure status of the system monitors, i.e., SSFD, computer output monitor, and servo monitor. From all of the available information, the maintenance update state will resolve the LRU location in which a failure has occurred and, if the failure has not already been registered, record it for future use by maintenance personnel in a non-volatile section of memory. The maintenance update state will then clear the in-flight test fault record and the deadline timer and transfer control to STP processing for another iteration around the test loop.

Module: ARCS.RELTIM.BAKGND.STPRO: System Test Panel Processing

- a) Description: STP input processing consists of checking for a device controller interrupt flag and, when it is set reading the corresponding request. Only two STP requests will be acknowledged during background tasks. One is a read request which will cause the current system status to be displayed, and the other is a ground test request which will be acknowledged only when aircraft status information indicates an "on ground" condition. If ground test is called for, program control will transfer out of background, out of real-time and into the ground test program. If the STP device controller interrupt flag is not set, control will immediately transfer to in-line testing.

STP output processing only gets executed following a read request received by the STP input processor. Under this circumstance the appropriate failure status information will be formatted and sent to the STP device controller for transmittal to the panel where it is displayed to the operator.

APPENDIX C

FUNCTIONAL LISTING OF MCP-701A INSTRUCTIONS

LOAD/STORE INSTRUCTIONS

Mnemonic	Instruction Description	Opcode	Instruction Format	Execution Time
LDU	Load UR from Program Memory	40XX	1	2.0
LDUS	Load UR from Scratchpad Memory	7CXX	2	1.5
LULB	Load UR (Left Byte) Immediate	00XX	4	1.25
LURB	Load UR (Right Byte) Immediate	05XX	4	1.25
LDL	Load LR from Program Memory	42XX	1	2.0
LDLS	Load LR from Scratchpad Memory	80XX	2	1.5
LLLB	Load LR (Left Byte) Immediate	01XX	4	1.25
LLRB	Load LR (Right Byte) Immediate	06XX	4	1.25
LDA	Load XA from Program Memory	46XX	1	2.0
LDAS	Load XA from Scratchpad Memory	88XX	2	1.5
LALB	Load XA (Left Byte) Immediate	02XX	4	1.25
LARB	Load XA (Right Byte) Immediate	07XX	4	1.25
LDB	Load XB from Program Memory	48XX	1	2.0
LDBS	Load XB from Scratchpad Memory	8CXX	2	1.5
LBLB	Load XB (Left Byte) Immediate	03XX	4	1.25
LBRB	Load XB (Right Byte) Immediate	08XX	4	1.25
LDC	Load XC from Program Memory	4AXX	1	2.0
LDCS	Load XC from Scratchpad Memory	90XX	2	1.5
LCLB	Load XC (Left Byte) Immediate	04XX	4	1.25
LCRB	Load XC (Right Byte) Immediate	09XX	4	1.25
STU	Store UR into Program Memory	4CXX	1	2.0
STUS	Store UR into Scratchpad Memory	94XX	2	1.75
STLS	Store LR into Scratchpad Memory	98XX	2	1.75
STAS	Store XA into Scratchpad Memory	A0XX	2	1.75
STBS	Store XB into Scratchpad Memory	A4XX	2	1.75
STCS	Store XC into Scratchpad Memory	A8XX	2	1.75

ARITHMETIC INSTRUCTIONS

Mnemonic	Instruction Description	Opcode	Instruction Format	Execution Time
ADU	Add to UR from Program Memory	4EXX	1	2.0
ADUS	Add to UR from Scratchpad Memory	ACXX	2	1.5
ADBU	Add to UR (Right Byte) Immediate	0AXX	4	1.25
ADBL	Add to LR (Right Byte) Immediate	0BXX	4	1.25
ADBA	Add to XA (Right Byte) Immediate	0CXX	4	1.25
ADBB	Add to XB (Right Byte) Immediate	0DXX	4	1.25
ADBC	Add to XC (Right Byte) Immediate	0EXX	4	1.25
AMS	Add to Scratchpad Memory From UR	B4XX	2	2.5
DIV	Divide UR & LR by Program Memory	58XX	1	10.75
DIVS	Divide UR & LR by Scratchpad Memory	C8XX	2	10.5
MPY	Multiply UR by Program Memory	56XX	1	6.0
MPYS	Multiply UR by Scratchpad Memory	C4XX	2	5.75
SBU	Subtract Program Memory from UR	52XX	1	2.0
SBUS	Subtract Scratchpad Memory from UR	BCXX	2	1.5

REGISTER INSTRUCTIONS

Mnemonic	Instruction Description	Opcode	Instruction Format	Execution Time
ABSU	Absolute Value of UR	19XX	4	1.25 - 1.75
CILB	Clear Indicator (Left Byte)-Immediate	1FXX	4	1.25
CIRB	Clear Indicator (Right Byte)-Immediate	20XX	4	1.25
CPLU	Complement UR	17XX	4	1.5
INV	Invert UR	16XX	4	1.25
SILB	Set Indicator (Left Byte)- Immediate	1DXX	4	1.25
SIRB	Set Indicator (Right Byte)- Immediate	1EXX	4	1.25
TSU	Transfer SR to UR	13XX	4	1.25
TUS	Transfer UR to SR	14XX	4	1.25
XUA	Exchange UR and XA	10XX	4	1.75
XUB	Exchange UR and XB	11XX	4	1.75
XUC	Exchange UR and XC	12XX	4	1.75
XUL	Exchange UR and LR	0FXX	4	1.75

INPUT/OUTPUT INSTRUCTIONS

Mnemonic	Instruction Description	Opcode	Instruction Format	Execution Time
CLR	Clear Device Controller	E8XX	2	1.25
CLRI	Clear Interrupt Specified	23XX	4	1.25
ENBL	Enable Interrupts from Device	ECXX	2	1.25
INHB	Inhibit Interrupts from Device	F0XX	2	1.25

SHIFT INSTRUCTIONS

Mnemonic	Instruction Description	Opcode	Instruction Format	Execution Time
SLZ	Shift UR Left-Enter Zeros	28XX	4	1.25 + .25(n)
SLZD	Shift Double Left-Enter Zeros	29XX	4	1.25 + .25(n)
SLZX	Shift Double Left by XC-Enter Zeros	2DXX	4	1.5 + .25(n)
SRC	Shift UR Right - Circulate Bits	2AXX	4	1.25 + .25(n)
SRCd	Shift Double Right - Circulate Bits	2BXX	4	1.25 + .25(n)
SRS	Shift UR Right-Repeat Sign	24XX	4	1.25 + .25(n)
SRSd	Shift Double Right-Repeat Sign	25XX	4	1.25 + .25(n)
SRSX	Shift Double Right By XC - Repeat Sign	2CXX	4	1.5 + .25(n)
SRZ	Shift UR Right-Enter Zeros	26XX	4	1.25 + .25(n)
SRZD	Shift Double Right - Enter Zeros	27XX	4	1.25 + .25(n)

DOUBLE PRECISION INSTRUCTIONS

Mnemonic	Instruction Description	Opcode	Instruction Format	Execution Time
ADD	Add Double from Program Memory	50XX	1	3.0
ADDS	Add Double from Scratchpad Memory	B0XX	2	2.5
ADMS	Add Double to Scratchpad Memory	B8XX	2	4.25
LDD	Load Double from Program Memory	44XX	1	3.0
LDDS	Load Double from Scratchpad Memory	84XX	2	2.5
STDS	Store Double into Scratchpad Memory	9CXX	2	3.0
SBD	Subtract Double from Program Memory	54XX	1	3.0
SBDS	Subtract Double from Scratchpad Memory	C0XX	2	2.5
ABSD	Absolute Value of Double Register	1AXX	4	1.25 - 2.25
CPLD	Complement Double Register	18XX	4	1.75 - 2.0
ZRD	Zero Double Register	15XX	4	1.5
NRM	Normalize Double Register	2EXX	4	2.0 + .25(n)

LOGICAL INSTRUCTIONS

Mnemonic	Instruction Description	Opcode	Instruction Format	Execution Time
NDU	And to UR from Program Memory	5AXX	1	2.0
NDUS	And to UR from Scratchpad Memory	CCXX	2	1.5
ORU	OR to UR from Program Memory	5CXX	1	2.0
ORUS	OR to UR from Scratchpad Memory	D0XX	2	1.5
CBSP	Clear Bits Specified by Bit Mask	78XX	3	2.75
SBSP	Set Bits Specified by Bit Mask	74XX	3	2.75
SKSP	Skip on Bits Specified by Bit Mask	70XX	3	2.0 - 2.25

BRANCHING INSTRUCTIONS

Mnemonic	Instruction Description	Opcode	Instruction Format	Execution Time
DSSZ	Decrement and Skip if Scratchpad is Zero	ECXX	2	2.5 - 2.75
JINT	Jump to Service Interrupt	21XX	4	8.0
JMP	Jump Unconditional	64XX	1	1.5
JMPI	Jump Unconditional, Indirect	66XX	1	2.0
JMS	Jump to Subroutine	68XX	1	1.5
JMSI	Jump to Subroutine, Indirect	6AXX	1	2.0
JSNS	Jump After Device Sense	F4XX	2	2.75
RTN	Return from Subroutine	2FXX	4	1.0
RINT	Return from Interrupt Routine	22XX	4	5.0
SIE	Skip If Program Memory Equal to UR	60XX	1	2.0 - 2.2
SISE	Skip If Scratchpad Memory Equal to UR	D8XX	2	1.75 - 2.0
SIG	Skip If Program Memory Greater than UR	5EXX	1	2.0 - 2.2
SISG	Skip If Scratchpad Memory Greater than UR	D4XX	2	1.75 - 2.0
SIL	Skip If Program Memory Less than UR	62XX	1	2.0 - 2.2
SISL	Skip If Scratchpad Memory Less than UR	DCXX	2	1.75 - 2.0
SKLB	Skip on Indicator (Left Byte) - Immediate	1BXX	4	1.25 - 1.5
SKRB	Skip on Indicator (Right Byte) - Immediate	1CXX	4	1.25 - 1.5
SKR	Skip if Device is Ready	E4XX	2	1.75 - 2.0

APPENDIX D

ARCS HARDWARE CONFIGURATION RATIONALES

This appendix contains the design rationale for the major configuration decisions involved with the candidate ARCS hardware architecture. In most cases, supporting rationale has been developed as a result of trade-off evaluations. In some cases, the rationale follows essentially from previous experience with fault tolerant digital systems. The following is an itemized listing of the key tradeoff areas considered, with a brief summary of implementation options:

- Processor Functional Characteristics
 - a) Advanced processor which is improved by DOT experience and other application evaluations.
- Electronics Packaging
 - a) Single LRU per system
 - b) Single LRU per channel
 - c) Separate computer and interface LRU's per channel.
- Cross-Channel Data Link
 - a) Autonomous operation in DMA mode at transmitter and receiver.
 - b) Processor controlled transmitter with DMA receiver.
 - c) Processor controlled transmitter with dual mode receiver - DMA for normal mode and interrupt driven for recovery mode.
 - d) Dedicated, one-way independent serial or parallel busses.
 - e) Non-dedicated, two-way serial or parallel busses.
- Level and Method of Synchronization (all options assume software sensor selection is required and is implemented)
 - a) Hardware methods resulting in bit identical, bit or frame synchronous processing.
 - b) Software methods resulting in bit identical, frame synchronous processing.
 - c) Software methods resulting in bit similar, frame synchronous processing - low gain equalization around path integrators required (following sensor selection).
 - d) Asynchronous processing - with equalization as in (c).

(cont'd)

- Watchdog Monitor
 - a) Simple analog pulse-width monitor
 - b) Precise digital timer
- Servo Monitoring
(all options assume the GE candidate servoactuator)
 - a) Independent, dedicated hardware monitors, with dual servo loop electronics per channel, which use cross-channel comparison and failure logic.
 - b) Same as (a) but no cross-channel comparisons in hardware form; all cross-channel servo monitoring in software.
- Non-Volatile Memory
 - a) Electromagnetic core
 - b) CMOS RAM with battery
 - c) MNOS memory

The discussion in the following subsections covers the major ARCS hardware architecture decisions in each of the listed trade-off areas.

1.0

PROCESSOR FUNCTIONAL CHARACTERISTICS

In evaluating the desirable and undesirable features of the MCP-701 central processing unit, the DOT experience and usage provided the judgment basis. The major undesirable features of the MCP-701 equipment were:

- Large overhead burden associated with interrupt initiated input output processing.
- Significant time required for DMA input output processing.
- Inability to use read only memories in place of core random access memories.

The desirable features of the MCP-701 related equipment were:

- A varied and powerful instruction repertoire, such as the one used by the MCP-701, has proved to be well suited to the computational needs of the real-time control.
- A 16-bit standard and 32-bit double precision word size in a fixed-point machine has been found adequate for control computation.

- Inclusion of dedicated hardware fault detection in critical functional areas (memory, arithmetic section, etc.) proved to be vital in demonstrating and maintaining correct system performance.
- Modularized, autonomous operation of the CPU and memory, CPU and I/O, and the I/O and memory, has been found to be both useful and efficient, especially for the purposes of automatic testing and fault isolation.
- Flexible input output system capable of dealing with both computer controlled and interrupt driven devices was a necessity.

A detailed analysis on instruction usage was conducted. The DOT applications indicated a typical control processor instruction mix as: Load/Store - 40%, Multiply - 3%, Divide - .5%, (other) Arithmetic - 5.5%, Logical - 5%, Branch - 13%, Non Memory Reference - 32%.

Using this mix to evaluate computer throughput shows a faster machine than as predicted by other often used mix equations. Figure D-1 shows throughput of the MCP-701 as predicted by three methods.

The detailed analysis work performed was of great benefits in suggesting changes to the architecture and organization of the MCP-701. The specific changes involved:

- Architecture/Opcode Modifications
- Memory Organization/Interface
- Input Output Organization/Interface

The opcode and architecture modifications were a direct result of the opcode usage survey. It was determined that:

- Use of multiple index registers would be very beneficial.
- Given the presence of multiple index registers, the usefulness of indirect addressing was limited.
- Displacement addressing (plus or minus) would be a more efficient instruction addressing mode. It would tend to make the page boundaries become transparent.

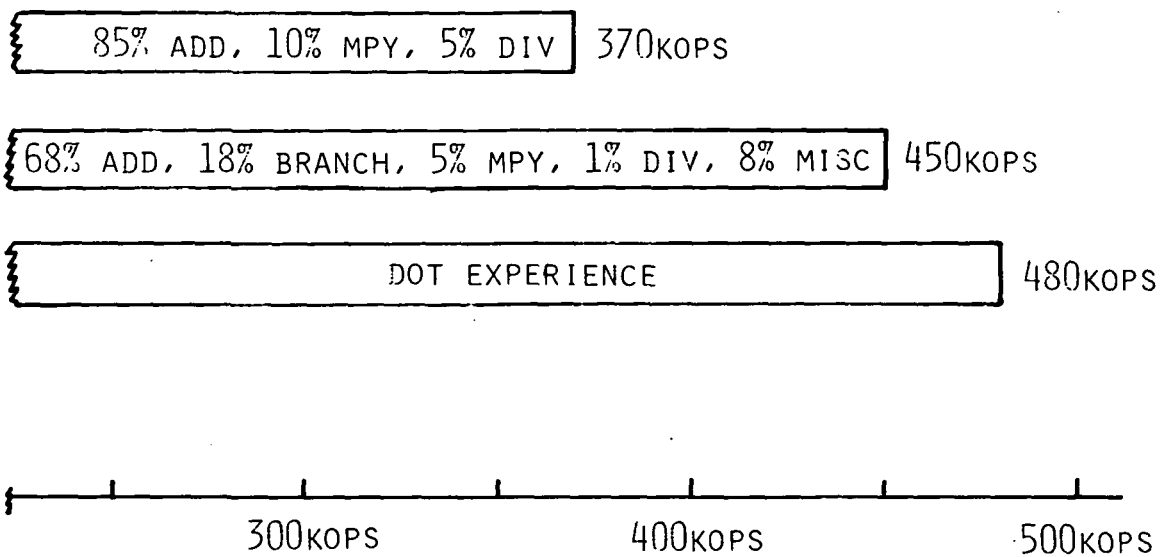


FIGURE D1? MCP-701 THROUGHPUT PREDICTION

- Little used or unused opcodes were eliminated from the instruction repertoire.
- New instructions were suggested for the instruction set (such as immediate instructions).
- "Macro" instructions were suggested to minimize the interrupt overhead burden.
- A more useful set of BIT manipulation instructions were suggested.
- A masking capability over all priority interrupt was suggested.
- The status register was expanded from 8 to 16 bits.

A new memory organization/interface was defined. It maintains independence between program and variable memory:

- A separate, independent addressing and control structure is provided for program and variable memories.
- Program memory may physically be in the form of semiconductor read only (ROM), programmable read only (PROM), or Core.
- Complete timing and addressing compatibility is provided between the various program memory forms.
- A mix of the program memory types is allowable.
- A semiconductor random access memory (RAM) is to be provided for variable storage.
- The semiconductor RAM is to be present regardless of the form of the program memory.

The above guidelines allow great flexibility in expanding the memory size. They allow a "zero software change" if PROM's or ROM's replace all or part of a Core memory. The separate program and variable memories eliminate all time burden associated with traditional direct memory access operations.

The input output organizational changes are consistent with the memory and CPU architecture modifications. It was determined that the input output structure be imbedded in the variable memory structure. The concept of the directly operable input output (DOIO) evolved. The input output (DOIO) has the features:

- ● Input output devices are addressed as if they were scratchpad memory locations.
- The CPU may perform arithmetic, logical, or branching operations directly on the I/O data.
- A significant increase in computational throughput results since data does not have to be software transferred to main line memory before manipulation.
- Input output devices may communicate with the CPU under processor control (PC), interrupt initiation (DVC), or direct memory access (DMA) modes of operation.

The overall suggested changes defined an organization which was modular and expandable. They defined an architecture which is consistent with various redundancy levels, is flexible to permit degraded redundant performance and is suitable for auto restart or recovery techniques.

The baseline ARCS processor was evaluated against other aerospace processors using a benchmarking technique. Improvements to the MCP-701 instruction set, addressing structure, and I/O system were systematically evaluated. The effectiveness of the ARCS processor is illustrated by the performance comparisons shown in Figures D-2 and D-3. Figure D-2 is a throughput comparison (using a conservative throughput mix equation), and Figure D-3 shows the input output processing time overhead burden. The combination of the directly operable input output (DOIO) with the high throughput of the processor makes this processor ideally suited to the needs of real time, redundant, recovery oriented, control applications.

The current implementation of the ARCS processor utilizes state of the art medium scale integration (MSI) and large scale integration (LSI) devices. They are low in power, currently available, and operate over the temperature and environmental range required for airborne applications. This implementation is what is expected to be used in a near term and intermediate term production aircraft.

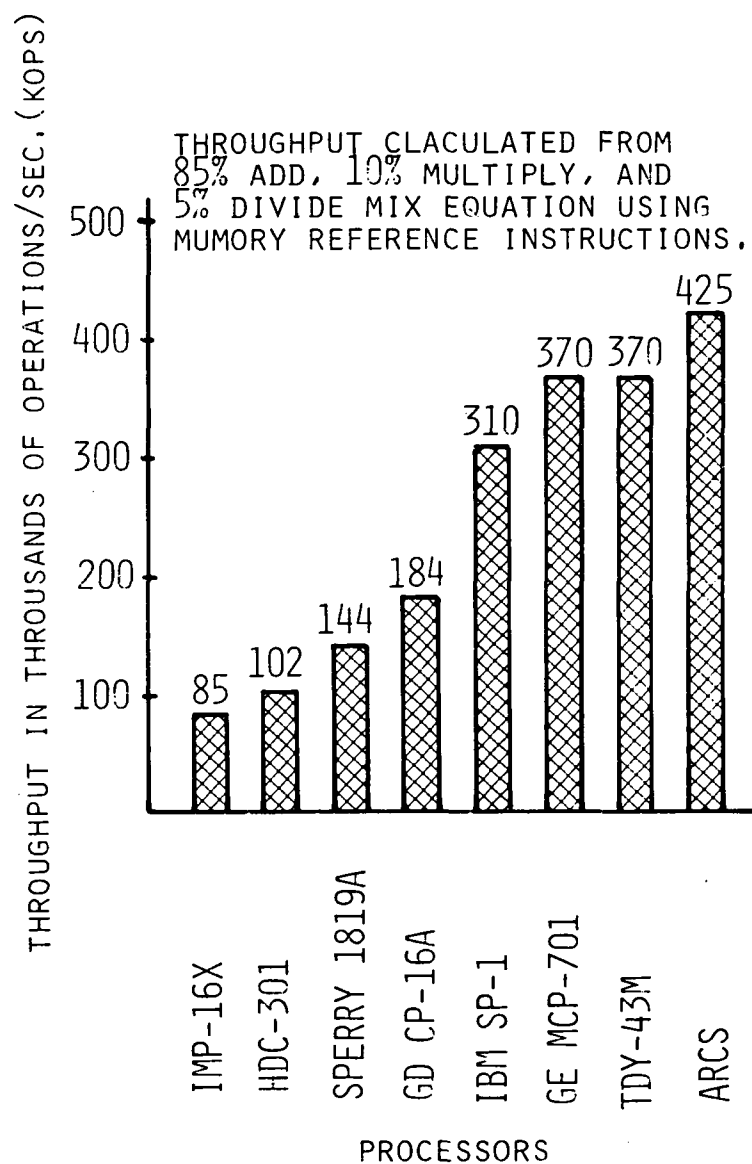


FIGURE D2 AEROSPACE PROCESSOR THROUGHPUT COMPARISON

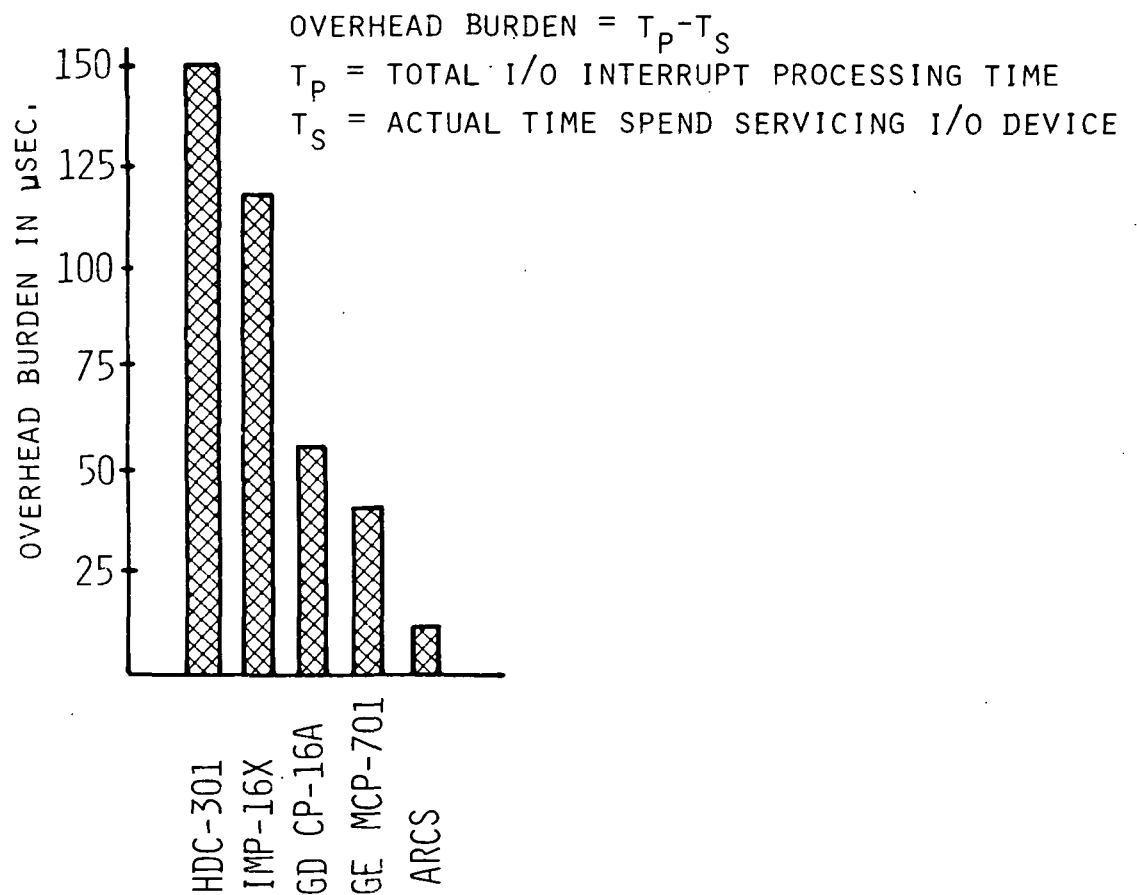


FIGURE D3 AEROSPACE PROCESSOR INPUT/OUTPUT OVERHEAD BURDEN

The current trends toward microprocessors and n-bit slice micro-controller technology are deemed useful for the far term production aircraft. In their current form such designs have the following limitations:

- Environmental temperature range
- Limited throughput
- Limited or non existent second source availability

It is anticipated that the far term ARCS implementation will use such devices. The functional definition and system architecture may remain substantially intact. Little or no change in redundancy configuration or reconfiguration strategy can at this time be predicted as a result of this new technology. The approaches developed in using the near term ARCS implementation will also be the basis for the far term implementation.

ELECTRONICS-PACKAGING

The possibility of packaging all ARCS electronics in a single LRU per system is rejected because:

- The single LRU would (at least in the near term) exceed standard ATR package size requirements for commercial transports.
- In flight-critical and flight-crucial applications the vulnerability of a single LRU to physical and electrical damage is inconsistent with flight safety requirements.

The major trade-off in the packaging area concerns a single LRU per channel versus separate computer and interface units per channel. The primary factors which influence the trade are:

- Maintenance and logistics (cost-of-ownership)
- Functional reliability
- Hardware design

Assuming that the single LRU achieves an overall complexity factor that is less than twice the complexity of either of the separate computer or interface units - at less than the sum of their costs - then the cost-of-ownership

trade should clearly favor the single LRU approach. Since this is a reasonable assumption, the motivation for considering separate computer and interface units must be based upon functional reliability or hardware design benefits, not cost-of-ownership benefits.

From a reliability standpoint, there is one potential advantage for the separate unit approach. If it is possible to achieve autonomous operation of the interface unit, and sensor selection is employed in the computer units, then there is a true voting node between the units (relative to system inputs). In order to achieve autonomous operation of the interface units - in a system where bit identical, frame synchronous processing is required - it is necessary to supply the interface units with frame sync information. Unless hardware or software voters are used in the interface units, it is impossible to transfer frame sync information from the computer units without creating functional dependence. The addition of hardware voters fixes the redundant structure, and consequently eliminates application flexibility. The addition of software voters requires the addition of a processor to the interface units.

Without developing quantitative reliability data, it appears that there is no meaningful reliability benefit which results from the separate unit approach, unless hardware voters are added. Such hardware is considered unacceptable under the ARCS development ground rules. Further, even if voters could be added for frame sync information, it is doubtful whether the net reliability gain, after adding a new power supply and timing structure for the separate interface unit, would be significant.

From the point-of-view of hardware design, the single LRU approach is by far the simplest approach functionally, and in terms of the amount of I/O hardware required. Duplicate power supplies and local timing structures are eliminated. The possibility of needing duplicate buffer memories is completely eliminated. Further, with the DOIO concept the total interface multiplexing function is effectively eliminated in the single LRU approach.

2.0

(cont'd)

Having all I/O data sources available on the computer bus structure, as RAM memory locations, is a major advantage of DOIO which greatly simplifies software.

For all of the reasons just discussed, the single LRU per channel approach was selected for the candidate ARCS architecture.

3.0

CROSS-CHANNEL DATA LINK TRADE-OFFS

One of the most important trade study areas for ARCS is the definition of the interchannel data link structure. This interchannel data link will be used under normal operation to exchange sensor data for selection, computed outputs and servo position data for monitoring and check words for data link integrity checking. It will also be used to exchange computer state vectors for recovery. It is important that the software involved in communicating through the link be minimal under normal and recovery conditions, and that the link provide a high enough data rate so that flexibility is allowed for developing executive strategy. At the same time, a very substantial portion of the computer electronics may be involved in the interchannel data link so careful justification of all requirements is mandatory.

3.1

DATA TRANSFER REQUIREMENTS

The cross-channel data transfer requirements involve two factors: the total number of distinct data words to be exchanged (without regard for timing), and the maximum number of data words which must be exchanged for any one minor cycle of computation. Based upon Boeing's application models and the currently available software estimates; the following list indicates the maximum number of distinct data words to be exchanged.

Normal mode SSFD	33 words
Recovery mode SSFD	429 words
Normal and recovery mode control	
law state vector	120 words
Test and miscellaneous	5 words
	<hr/>
Total	587 words

These requirements reflect the use of dedicated sensor interfaces and a particular Boeing-derived sensor selection and failure detection (SSFD) algorithm. During each minor cycle of computation, only those words needed for that cycle must be exchanged cross-channel. Different combinations of data words may be exchanged during each minor cycle according to the solution rate requirements for the various program segments. The minor cycle time is currently planned to be 10 msec (corresponding to 100 solutions per second). While the exact executive time scheduling has not been determined, it is not expected that the rate of cross-channel data exchange will exceed 64 words in one minor cycle.

At the currently established data rate of $19.5 \mu\text{sec/word}$, the transmission of 64 words would require 1.28 msec. Even with the software overhead associated with loading the transmitter LIFO, the entire 64 word transmission would be completed in approximately 1.5 msec. With appropriate executive scheduling within a minor cycle, it is possible to overlay other functions during the actual transmission of data. Since data is received and stored in a DMA mode at each receiver, the relative time burden in a 10 msec minor cycle may be made minimal (as little as 0.25 msec).

The provision of a 10-bit address label on each data word, and a corresponding 1024 word buffer memory at each receiver, is more than adequate to handle the 587 word requirement.

CROSS-CHANNEL I/O ARCHITECTURE CONSIDERATIONS

In formulating the structure for the cross-channel data transmitter and receiver, the major constraints to be satisfied were:

- To provide a cross-channel communication capability which would satisfy all of the normal mode exchange requirements yet minimize the transmitter and receiver overhead burdens associated with the cross-channel process. This capability must be such that bit identical sensor selection and output selection can be accomplished.

- To provide a cross-channel communication capability which would satisfy the large data requirements potentially required for a recovery process. The CPU burden for the transmitter must again be minimized such that it is not required to halt real time computation. It is assumed the receiver CPU burden is insignificant since it is in a recovery process.

The data requirements such that bit identical selection of sensor and output signals might be accomplished were discussed in Section 3.1. The desire to minimize the CPU burden for both the transmitter and receiver during normal operation and for the transmitter during a recovery operation will be discussed in this section. Three techniques were examined for minimizing CPU burden. They were:

- Direct Memory Access (DMA) operations
- Processor Controlled (PC) operations
- Use of MCP-701A application dependent "macro" opcodes.

The results are presented below.

3.2.1

Elimination of CPU Burden During Transmission

A concept to provide great flexibility would be to have the cross-channel transmitter as a processor controlled device. CPU instructions would then be required to transfer the data from one I/O zone to the cross-channel transmitter. If a looping subroutine were used for this purpose, and 64 variables were to be transmitted, the software routine would consume about 1.2 milliseconds. This is certainly not minimizing CPU burden.

The selected concept utilizes an MCP-701A application dependent opcode. It is a solution which minimizes the CPU burden yet allows the CPU great flexibility in selecting the source and quantity of data to be cross-channelled. This "macro" opcode would be defined as:

Mnemonic:	MOVE - Block Transfer Scratchpad Memory
Description:	Data is transferred in a block of XC words from one scratchpad memory zone to another. Index register

B (XB) specifies the source address for the first word. Index register A (XA) specifies the destination address for the first word. Index register C (XC) specifies the number of words to be moved in a block. Data is transferred in order of descending address. Thus, the source zone corresponds to locations XB through XB - XC, and the destination zone corresponds to locations XA through XA - XC. There are no address mode variations allowed with this instruction.

When used to load the LIFO in the cross-channel transmitter, the MOVE instruction is used as described in subsection 3.2.3.1.3. The execution time for the instruction is determined as follows:

$$ET = 1.5 + 2.5 (XC) \mu \text{ sec}$$

For a block transfer of 64 words into the LIFO transmitter, the execution time is 161.5 μ sec.

The MOVE opcode reduces the processor overhead burden by more than 7:1 for cross-channel transmissions. Further, it minimizes the hardware complexity of the transmitter. Flexibility is inherent in the opcode since the programmer may select the block size and starting address. A series of MOVE instructions may be programmed to select data from multiple source zones.

3.2.2 Elimination of CPU Burden While Receiving

The primary consideration here involves the receipt of recovery mode data as opposed to normal mode data. Within the DOIO structure of the MCP-701A, the DMA mode of operation for the receiver is the clear choice to minimize CPU burden. However, the amount of data required in the recovery mode may exceed the size of the local scratchpad memory dedicated to each receiver. To handle this problem, one approach would be to allow the receiver to revert to an interrupt initiated processor controlled mode.

3.2.2 (cont'd)

This approach has two disadvantages; one of which is judged to be fatal. First, the CPU burden is increased during recovery, but this is of minor importance, because the CPU has nothing else more pressing to do than recover. The more serious difficulty is that using an interrupt initiated receiver mode requires that Boeing requirement (c) in subsection 2.3 be violated. That is, one computer channel must have the capability to interrupt another. Because this is not allowed on ARCS, the decision was made to utilize a 1024 word scratchpad RAM at each receiver in a DMA mode. The CPU receiver burden is therefore zero, and no cross-channel interrupts are needed.

3.2.3 Data Rate and Format

As indicated earlier, the selected cross-channel transmission rate is one word every 19.5 μ seconds. This rate is based upon the bit serial transmission of 29 bits (16 data, 10 label, 2 sync, 1 parity) at a 2 MHz rate with 5.0 μ seconds word separation.

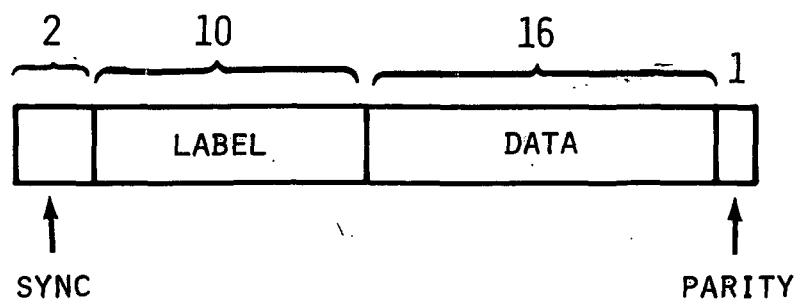
The word format is illustrated in Figure D-4. The signal format is a self-clocking format derived for MIL-STD-1553. This signal format requires only two wires per transmission line, and is compatible with AC coupled media, such as certain fibre optic transmission systems employ.

3.3 SERIAL VERSUS PARALLEL TRANSMISSION

The considerations here involve:

- Hardware complexity and thus reliability.
- Data rate requirements such that a bit identical selection process may occur.
- Recovery strategy
- Use of dedicated vs non-dedicated bus structures.

Section 3.1 discussed the data exchange requirements. A word serial, bit serial transmission format is believed to be adequate for the indicated requirements. The transmission of 64 variables would require only



WORD FORMAT

FIGURE D4 CROSS-CHANNEL WORD FORMAT

1.25 milliseconds. As little as 0.162 milliseconds of CPU burden may be associated with the transfer process as pointed out above. This allows the CPU to pipeline or overlay other software executive functions.

Reliability considerations are obviously in favor of the serial transmissions. Fewer parts are involved. Cabling and connector cost, and aircraft weight considerations all favor the serial transmission format.

The bit and word serial format is consistent with the recovery considerations and strategies presented below.

The trade-off concerning the use of independent, dedicated one-way busses, versus any non-dedicated bus structure may be resolved, in most practical implementations, by simply considering cable failure effects. In particular, the effects of cable failures on sensor selection are of critical interest.

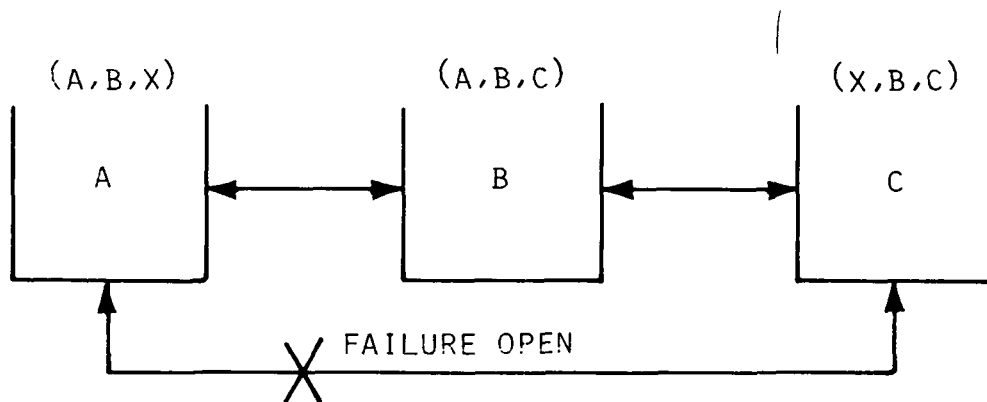
Figure D-5 illustrates the common type of "open" failure for two cases.

In case 1, the cable failure shown is a "single-point" failure whenever bit identical sensor selection is employed. The reason for this is that with the two-way busses a single "open" failure causes each channel to sensor select from a different set of three sensor values (as shown in Figure D-5). The independent, dedicated one-way bus structure shown as case 2 does not have this problem. For this reason, and other similar failure effect reasons, the case 2 bus structure has been selected for ARCS.

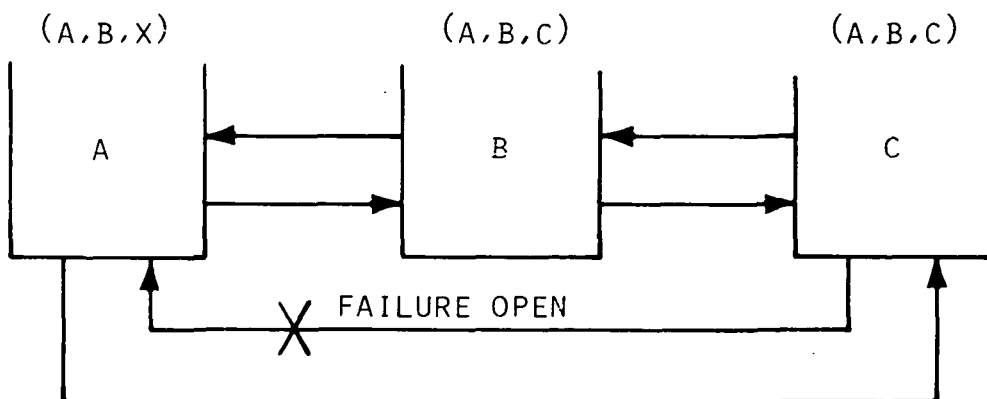
4.0

SYNCHRONIZATION TRADE-OFFS

One of the most basic decisions to be made in the design of a redundant digital computing system such as ARCS concerns the level and method of synchronization to be employed. Traditionally, this decision has involved one or more of the following three considerations, and has been intimately related to decisions concerning sensor selection and computer output monitoring:



CASE 1 - NON-DEDICATED TWO-WAY BUSSES



CASE 2 - INDEPENDENT, DEDICATED ONE-WAY BUSSES

FIGURE D5 CABLE FAILURE EXAMPLES

- a) A desire to hold the input sample times and output update times for corresponding signals in each channel within a specified tolerance of each other.
- b) A desire to eliminate all drifting of path integrators implemented in the control laws and thus obviate the need for any cross channel equalization.
- c) A desire to have bit identical computer outputs for corresponding signals in each channel.

These considerations are closely interrelated and involve a whole spectrum of tradeoffs in both hardware and software areas. For example, if bit identical outputs are desired, frame synchronization with sensor selection or cross-channel equalization is mandatory. Either hardware or software or mixed implementations are possible for all of these functions. To compound the tradeoff problem, the question of synchronization for the ARCS system involves the further consideration of the requirements for recovery.

The following discussion considers various tradeoffs with the important simplifying assumption that whatever level of synchronization is employed, it is a software process. It is the recovery of a faulted computer to operational status which has the largest potential impact upon the level of synchronization required.

Several different recovery procedures have been hypothesized for providing transient fault tolerance within redundant computing systems. Each of these recovery procedures addresses a certain class of transient faults and carries implications for normal system operation in many areas. Certain of these recovery procedures require substantial additional hardware beyond that required for a nonrecoverable system, which leads to a lower basic channel reliability, and thus, limits the practical advantages of incorporating them. The following subsections detail the hardware and software considerations involved in implementing the five recovery procedures entitled roll ahead, roll back, memory copy, restart, and coasting.

4.1

SYNCHRONIZATION AND RECOVERY STRATEGY

Program roll ahead is defined by ultrasystems to be capable of effecting "instantaneous" transient fault recovery under certain conditions. According to the definition, roll ahead involves the exchange of state variables between each pair of redundant computers following the execution of each program segment. Each program segment operates on an input state vector to generate an output state vector, which is then used as the input state for the next program segment. If the program memory is intact in the computer which suffered the transient fault then recovery is possible.

Recovery would simply involve having the faulted computer use the state vector from an unfaulted computer for the next and possibly several ensuing program segments. However, the failed computer must be notified by another computer (ultrasystems suggests an interchannel interrupt) to use an unfaulted computer's state variables for subsequent program segments, as needed, until the faulted computer's state vector agrees with those of the unfaulted computers.

There are a substantial number of underlying assumptions behind the ultra-systems roll ahead procedure, a few of which are

- 1) An interchannel data link structure exists between each pair of computers which is capable of very high data exchange rates.
- 2) The application program is carefully segmented.
- 3) An unfaulted computer channel can be trusted to interrupt the processes of a faulted computer in a predictable manner and that spurious interrupts generated by a faulted computer can be properly inhibited from affecting the normal operation of an unfaulted computer.
- 4) The variable portion of each computer memory is n (n -tuple redundancy) times the size required for simplex operation, or, that the computers are running in a frame synchronous manner where each frame defines a program segment.
- 5) Each computer verifies its program segment state vector against all other computers' vectors at the end of each program segment and before proceeding to the next segment.
- 6) There are relatively few elements in the state vector for each program segment, or, that all computers are operating on identical data in a frame synchronous manner, and are expected to produce the same state vector, so that the comparison execution time is minimized.

For the ARCS, assumption 1 above would dictate the inclusion of an inter-computer parallel data bus structure. This has been judged to require too much electronics for the ARCS. Assumption 3 violates a specific ARCS reconfiguration principle ruling out such interchannel interrupts. Assumption 4 significantly increases the size and power requirements for each computer in such a system. Assumption 2 is typical of real time flight control problems, and assumptions 5 and 6 fit easily within the anticipated operational scenario of the ARCS.

The basic notion of roll ahead, state vector exchange between an unfailed computer and one which has been judged to be transiently failed, is a sound procedure, and the ARCS has been defined to use this procedure in a modified version as one of the primary recovery mechanisms.

Roll back simply involves the recomputation of a specific program segment if the interchannel state vector comparison operation at end of that segment indicated an error in that computer. Roll back as defined by ultrasystems involves all of the assumptions indicated under roll ahead. In addition, the use of roll back as a recovery procedure dictates that all application program segments be written so as to preserve all state vector values, both from the previous iteration and from the current iteration. This causes an effective doubling of the size of the variable memory. Roll back is primarily applicable to a relatively restricted set of transient faults, i.e., those where only the computed results are corrupted. Roll back is primarily useful as a transient fault isolation/recovery procedure when only two computers have been operating satisfactorily and a difference is detected between their results, but no fault isolation is available from other sources. The ARCS hardware has been configured to permit a roll back recovery procedure to be used provided that appropriate procedures for use of the relatively low rate interchannel data transfer are adopted.

Memory copy involves the exchange of the program and variable memory contents between an unfailed computer and a failed computer in an effort to recover from transient faults in the memory electronics which have permanently altered the contents of the program memory. The ARCS has been defined to use ROM or PROM program memory so there is no requirement for such a recovery procedure.

Restart simply involves re-initializing the program variables and developing a new program solution upon detection of an out-of-tolerance condition. The ARCS has been defined to make use of this recovery procedure. Flight control problems are closed loop computations and as such contain information from a relatively short period of history (typically less than 2 minutes). That is of course an impossibly long time for recovery of a computer unless other unfailed computers are satisfactorily controlling the aircraft. It does have the advantage that the operation of the unfailed computers need not be disrupted while the failed computer is recovering. It also dictates that something less than bit-for-bit identical results be established as the monitoring criteria at the computed outputs.

RECOVERY IMPACT ON NORMAL OPERATION

As indicated in the foregoing discussion of the various potential recovery mechanisms, the inclusion of recovery as part of the system operational scenario has a significant impact on many hardware and software system characteristics. The ARCS recovery mechanism is a modified roll ahead where the program segment output state vector is transferred across channel throughout the computation of a program segment rather than waiting until the end of the program segment computation. This approach is more compatible with the LIFO driven cross-channel transmitter which is used with the ARCS system than it would be with a conventional DMA mode cross-channel block transfer. In addition, it permits the use of serial transmission with the data rate of 19.5μ sec per word as defined for the ARCS. This recovery

4.2

(cont'd)

procedure does not dictate any level of synchronization for the ARCS since sufficient storage is allocated within the cross-channel receiver for unique storage of all interchannel data including the recovery data.

The ARCS uses a frame synchronized computational scheme with all computers performing the same computation during each frame. Since the same sensor data is used by all computers under normal operation (on executive strategy decision), this approach yields bit-for-bit identical computer outputs.

4.3

SOFTWARE FRAME SYNCHRONIZATION

Two basic algorithms to perform the synchronization task were investigated. The first algorithm waited for all computers to be ready; the second algorithm selected one computer as the master and the others followed it. To prevent any possibility of the system hanging up on first failure, all the wait loops for both algorithms have a time out counter included in their mechanization.

In all of the methods studied, defined by the table of Figure D-6, the three sync discretes are assumed to be logic indicators L1, L2, and L3 in the status register of the CPU. Each computer can set one of the indicators and test all three to monitor the state of the computers. Two schemes for interconnecting the indicators were investigated. In the first interconnection scheme all computers set L1, but tested only L2 and L3. That meant that the meaning of L2 and L3 was different in each channel. The interconnection of the indicators was permuted in the wiring harness. This interconnection results in the least software to set, clear and test the sync commands. The second scheme did not permute the meaning of the indicators. Each computer determined which slot it was plugged into, and set either L1, L2 or L3. This resulted in the corresponding indicators in the other channels being set.

	PERMUTED SYNC INDICATORS	EACH SYNC DEDICATED TO A CHANNEL	WAIT/RELEASE SYNC INDICATOR	MASTER/SLAVE SYNC ALGORITHM	RE-EVALUATE FAILURE STATUS	LATCH SYNC FAILURE STATUS	TEST SYNC FOR BOTH INDICATORS	AS LSC SET & CLEAR	FAILURE DETECTION ON WAIT AS FSC	FAILURE DETECTION BASED TEST OF TIME FOR FSC SET	FAILURE DETECTION INCLUDES MEMORY DETECTION ON EXIT	IN WORDS	MAXIMUM REQUIREMENTS	MAXIMUM SKEW IN MICRO SECONDS
METHOD I	X		X		X		X		X	X		35	4.4	
II		X	X			X	X	X	X	X	X	140	8.4	
IV		X	X		X		X	X	X	X	X	114	4.4	
III		X		X		X	X	X	X	X	X	110	3.0	

FIGURE D6 SUMMARY OF CANDIDATE SYNC ROUTINES

The synchronization performance for a particular implementation of an algorithm is measured by the SKEW between the three channels. SKEW is defined to be the difference in time when any two channels reach the SYNC point in a particular implementation. Worst case SKEW would be the largest SKEW between any two channels.

Because of the wait loops in the mechanizations described, each computer will detect that another is ready for synchronization some variable amount of time after the LSC is set in the other computer. The time will depend upon where the local computer was in the wait loop at the time the LSC was set, but is bounded by a minimum and maximum value. Looking at Figure D-7 let T_0 be the time when the sync algorithm is satisfied. Some time t_1 later is the shortest time that a computer can respond and get to the sync point for a particular implementation. The time t_2 would be the maximum time for which it can be stated with absolute certainty that the indicator will be detected. The maximum SKEW = $t_2 - t_1$ would be defined by t_2 of the latest of the three computers and t_1 of the earliest to arrive at the sync point.

The first sync algorithm, the "Wait" approach, requires that all computers (for no detected failures) be ready before sync release is achieved. Figure D-8 is a simplified flow chart for this algorithm, which has the following characteristics:

- Set LSC
- Wait, testing the FSC, until one of the other computers is ready.
- If neither is ready before time limit is exceeded, interpret this as loss of sync.
- After second channel is detected, wait for third channel.
- If time limit is exceeded for third channel, mark it failed and continue to SYNC point.
- Clear LSC.
- Test sync indicators to be sure none failed set.

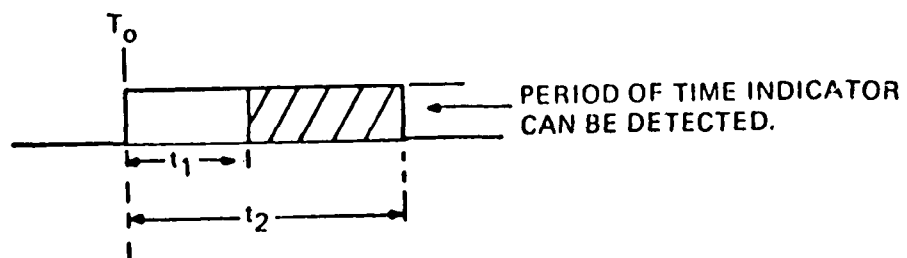


FIGURE D7, SKEW DEFINITION

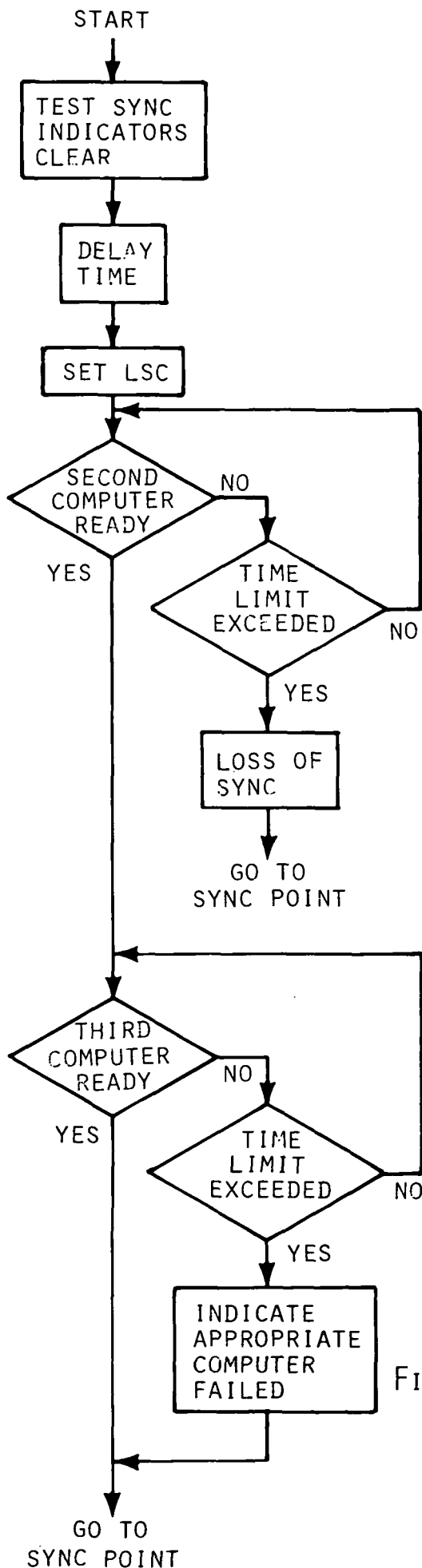
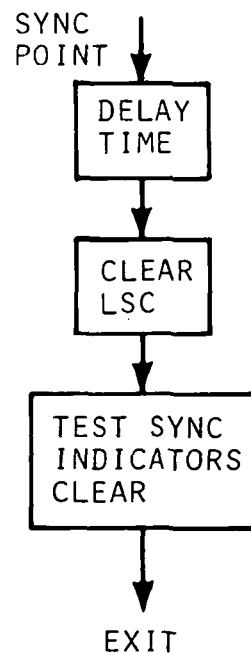


FIGURE D8 THE "WAIT" ALGORITHM FOR FRAME SYNCHRONIZATION



The second sync algorithm, the "Master" approach used the concept of a master computer, as shown in Figure D-9 and has the following characteristics:

- Test sync indicators for clear.
- Set LCS.
- Test sync indicators for set.
- Select master computer based on the failure information available.
- If local computer is master, wait for a time equal to maximum allowable SKEW, then clear LSC.
- If local computer is not master, go into a wait loop testing master computer's sync indicator until it is cleared.
- Test sync indicators to be sure none failed set.

The relative merits of each approach become more apparent when the practical consideration of failure detection is discussed.

The primary requirement for sync routine failure detection is that loss of synchronization be detected. In addition, information can be collected that the local channel is a sync with only one other channel. This information is then passed on to the executive. If a local failure was detected, the executive may wait until the error has been detected in several consecutive frames before disengaging the servo actuators.

Figure D-10 shows the real time "window" that defines the period of time within which all channels should set their LSC. For the cases where one or more computers are outside that "window", there is no information available in the local computer to allow it to detect the failure. The only real time information available to the local computer is that it has had a timer interrupt. Since the three clocks are not perfectly synchronized, this occurs at slightly different times in each computer.

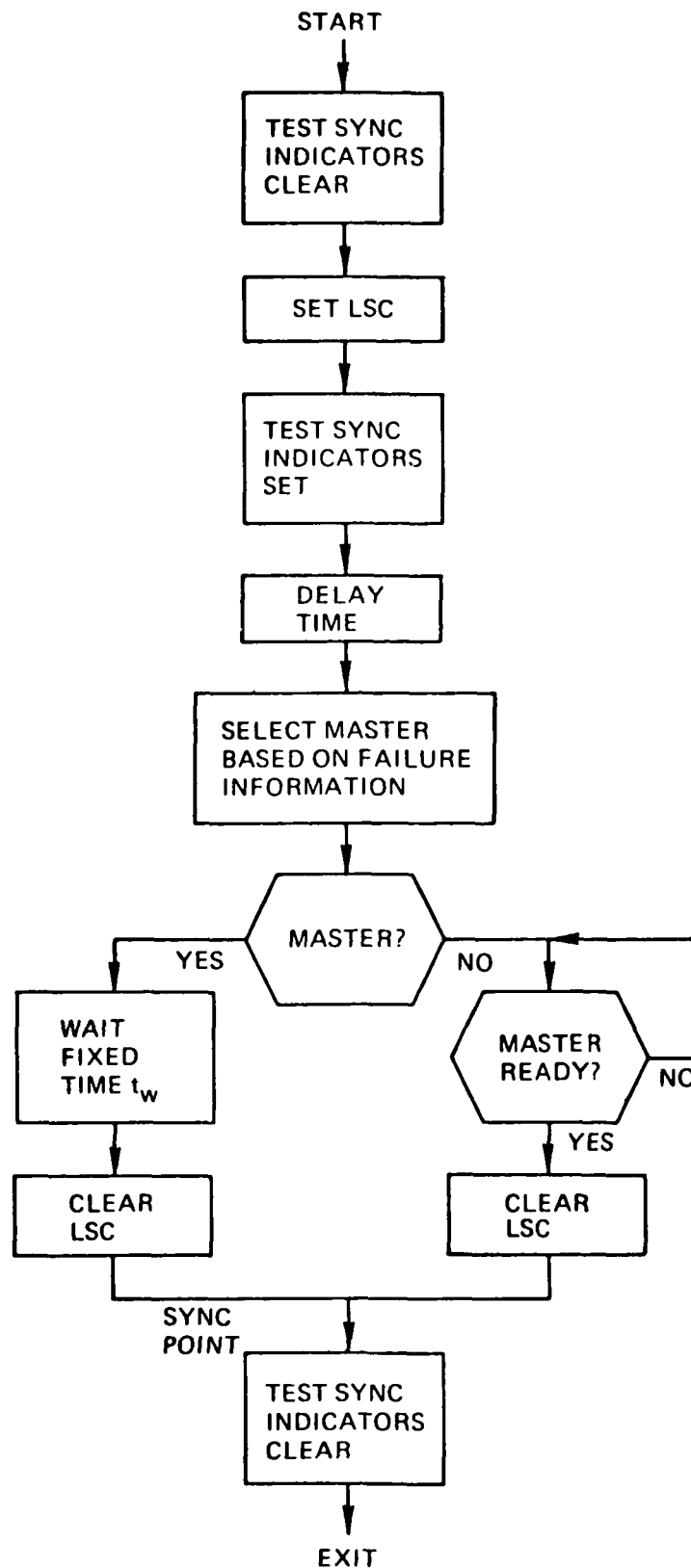


FIGURE D9 THE "MASTER" ALGORITHM FOR
FRAME SYNCHRONIZATION

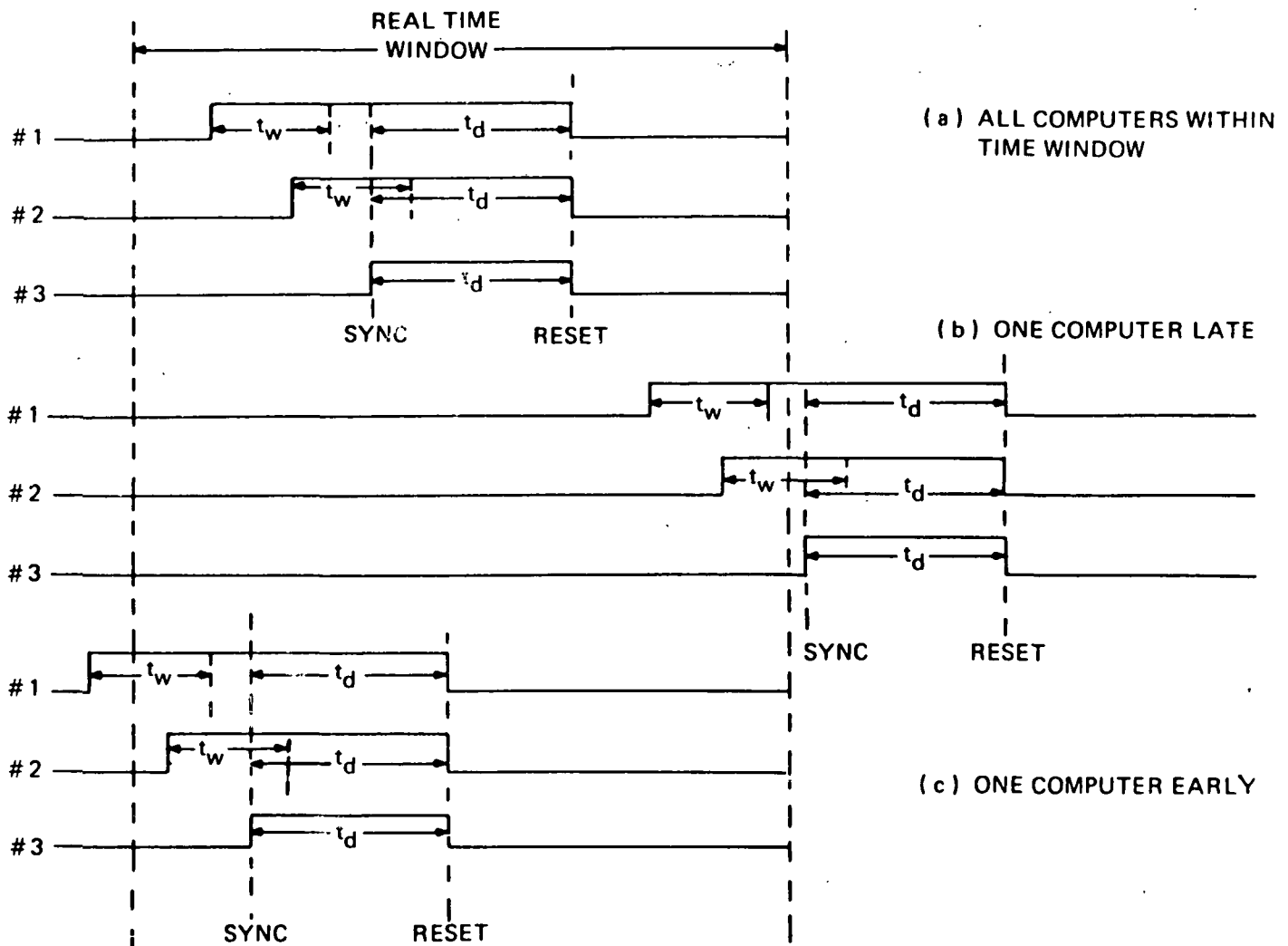


FIGURE D10 "TIME WINDOW" POINT OF VIEW FOR FAILURE DETECTION

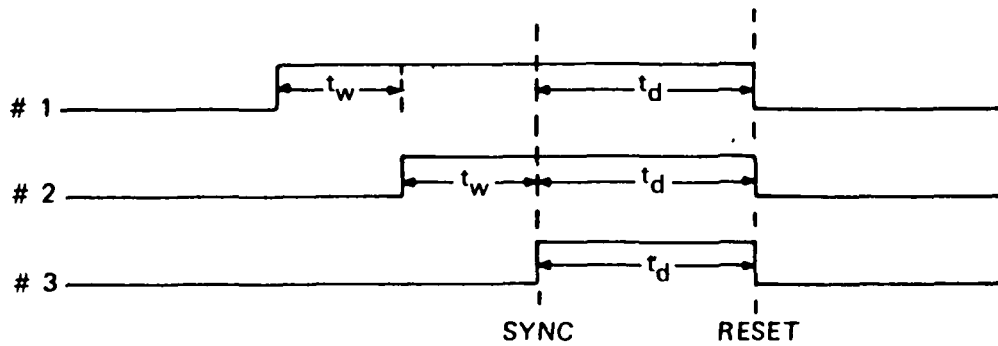
25/4

The method of failure detection proposed for the "Wait" algorithm is to define a maximum period of time during which each computer waits for the other two. If they are not detected ready to sync, the local computer indicates a failure. This time period must take into account tolerances on the three clocks as well as variations in software execution time due to branching decisions.

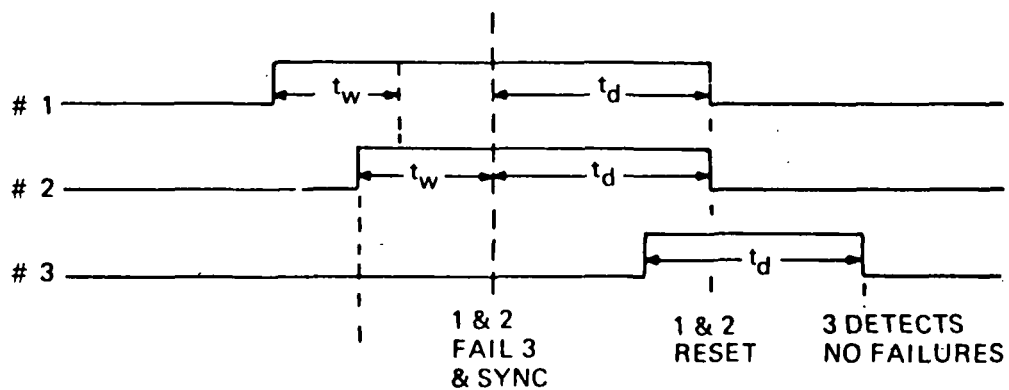
Figure D-11 shows a mechanization where all computers set their LSC, then wait a period t_w to see if the FSC's are set. A maximum difference of $2t_w$ can exist between the time the first computer sets its LSC and the time last computer sets its LSC, for the case of no detected faults. This time is twice as long as the tolerance specified for a single computer because the wait loop for both computers is re-initialized to t_w after the second computer is detected ready to sync. Both computers must wait for the same period of time for the last one in order to minimize SKEW.

The time t_d is the time the LSC is set during the sync routine after the computers have synchronized. This is a fixed time in all computers. If the third computer arrives after the maximum wait time of the other two, the first two will call the third failed. However, if the LSC of the first two computers is still set when the third checks, the third channel will detect no errors. All channels would detect the error if the FSC of the first two computers was clear when the third channel checks.

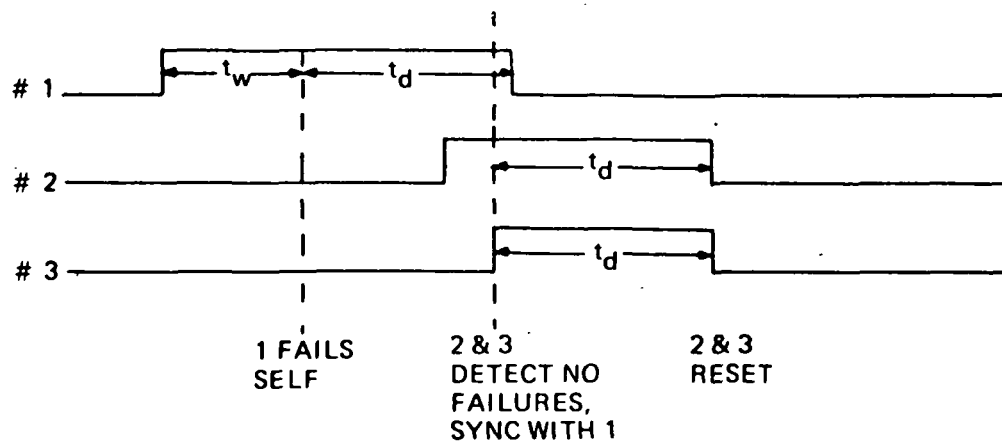
If the first channel is ahead of the others by more than t_w , it will detect two failures. In all of the mechanizations studied, two simultaneous failures are not considered probable. A simultaneous failure is defined as two computers failing within approximately one frame time. If the local computer detects neither of the other channels ready for synchronizations with no previous failures, it assumes itself failed. The other two channels will detect no errors if the LSC of the first channel is still set when they check.



(a) ALL COMPUTERS JUST UNDER MAXIMUM TOLERANCE



(b) COMPUTER #3 LATE



(c) COMPUTER #1 EARLY

FIGURE D11 FAILURE DETECTION BASED EXCLUSIVELY ON WAIT TIMES

This shows why the time t_d should be as short as possible in this mechanization. If it had been about half the time shown, all computers would have detected the failures in both cases.

The failure detection capability can be improved so that the failures shown in Figure D-11 are detected. This can be done by adding a test at the beginning of the sync routine to check the FSC's clear before setting the LSC. The local computer must wait $2t_w$ after checking the FSC's and before setting its LSC. By waiting for this time period, all computers within tolerance will be guaranteed not to set their LSC before the latest one checks it for clear.

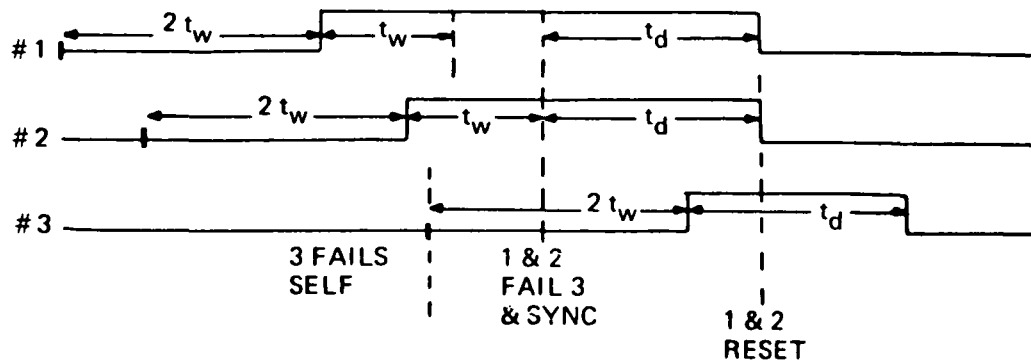
From the point the indicator is set, the algorithm remains unchanged.

Figure D-12 shows the last two cases in Figure D-11 with the addition of this test. In both cases, the test occurred when the FSC was high. The last case in Figure D-12 shows that all failures are still not detected identically in the three computers.

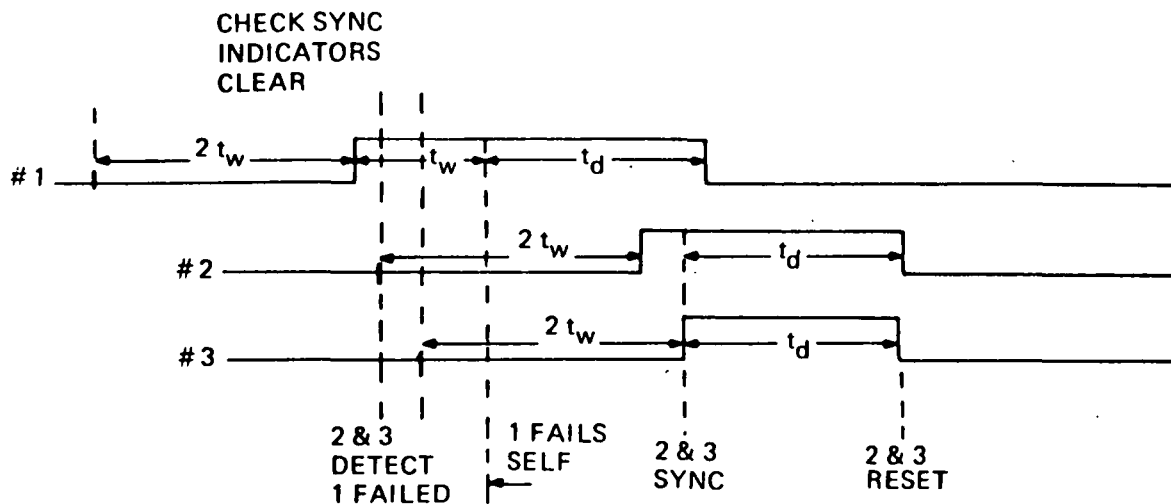
This illustrates that it is extremely difficult to guarantee unambiguous failure detection in all computers. For this reason, the test of the local computers sync command as well as the foreign computers indicators is included in each computer. Also, the redundant test of the sync indicators being cleared helps. The additional information tends to increase the probability of detecting failures.

Accurate failure detection is a basic requirement of the sync routine using the concept of a master computer. The master is selected based on whether channel #1 or #2 has failed, so all computers must make the same decision.

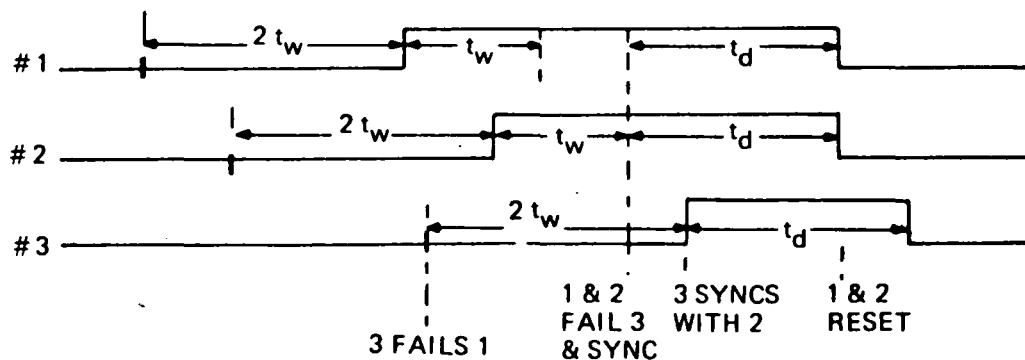
The ideas of failure detection developed for the "Wait" algorithm can be applied to the master computer approach also. A test of the sync indicators clear at the beginning and end of the routine will be assumed. Figure D-13



(a) COMPUTER # 3 LATE



(b) COMPUTER # 1 EARLY



(c) NON IDENTICAL FAILURE DETECTED

FIGURE D12: ADDITIONAL FAILURES DETECTED WITH TEST FOR SYNC INDICATOR CLEAR AT BEGINNING OF ROUTINE

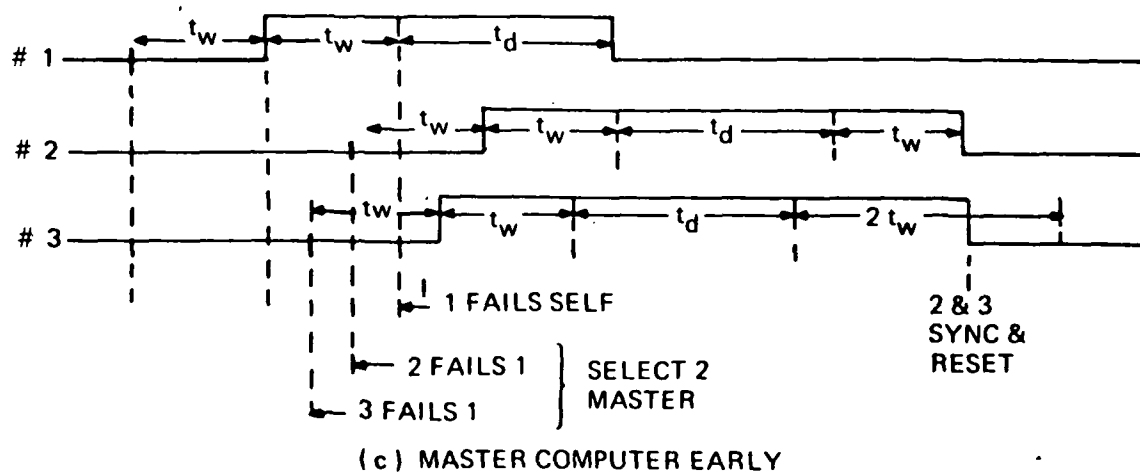
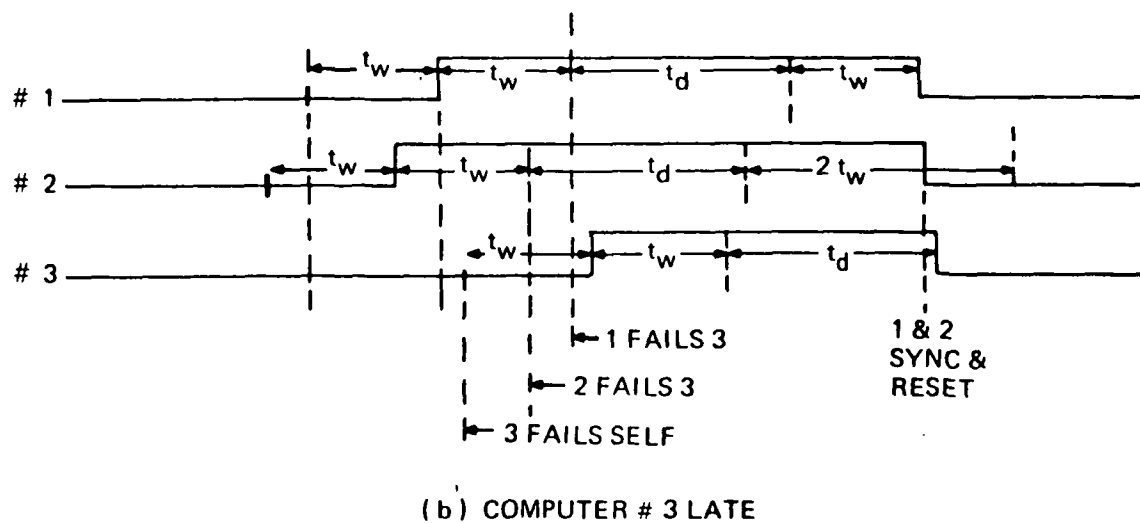
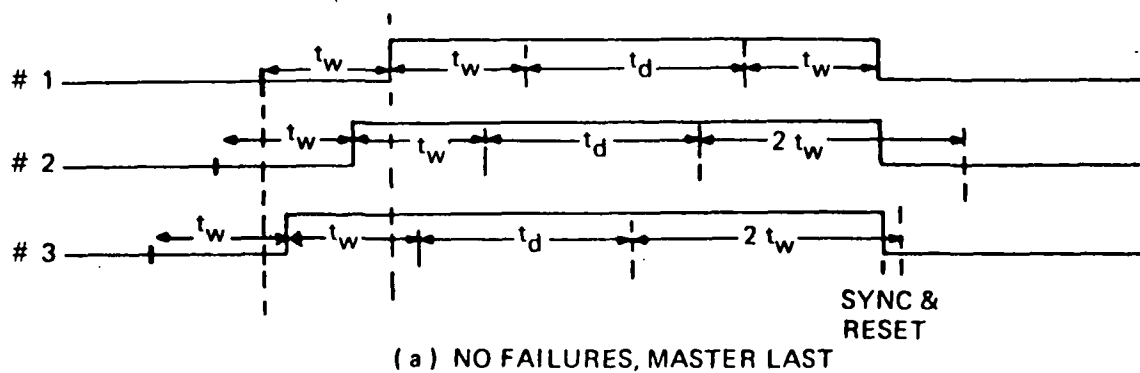


FIGURE D13 IDENTICAL FAILURES DETECTED IN ALL COMPUTERS FOR MASTER APPROACH

shows a case where the master computer is the last to set its LSC. All of the computers are within tolerance, but the maximum deviation between computers is one-half of what it was for the other methods. This is because all computers must arrive within the wait time t_w defined by the master computer. Each computer checks to see the sync commands are clear, waits t_w and sets its LSC. They then wait another period t_w and check to see the sync indicators are set. A time period t_d expires, then the master computer waits for a time t_w to allow all other computers to get into a loop testing the master's sync command. It then clears its sync command and continues. As can be seen in Figure D-13 the computers which are not master must wait for a time period of $2t_w$ for the master, in order to allow for the possibility of being early. Synchronization is achieved on the trailing edge of the LSC, exactly as the iteration timing reference is being reset.

Figure D-13 also shows a case where computer #3 is slower than the master by more than t_w . Because of the tests of the sync indicator being clear and set, all computers detect the failure. It also shows the master arriving before the other two by more than t_w . This failure was detected in all computers, which required #2 and #3 to select #2 as master. The sequence of events that results in detection of different failures in each computer is possible with the "Master" computer approach also.

All of the frame synchronization methods described here perform the sync task adequately, using the candidate hardware configuration, when all computers are within tolerance. One of the primary tasks defined at the start of the study was to determine the maximum SKEW for the various routines. It was believed that this would be very instrumental in selecting the synchronization routine. After doing a detailed analysis of the three methods described, the SKEW problem was found to be virtually nonexistent. The tolerance on the oscillators used for the iteration timing reference is 0.01%, which is a deviation of only 1 μ sec for a 10 millisecond frame time.

Therefore, synchronization is required to prevent only long term drift.

All of the sync routines described will hold the SKEW to less than 10 microseconds for no failures.

As the study progressed it became clear that detecting failures identically in each computer was the most difficult problem. All of the methods described allow some period of time where one computer may detect another failed, but the third will not. The sync routine just tries to detect those failures that prevent the local computers from synchronizing with one or more other channels. The overall failure detection task is the responsibility of the system executive software. If simultaneous failures are considered possible, then the executive must be very careful not to disengage the third channel when loss of sync is detected with no previous failures. This would result in loss of the system with two failures.

After further study it may be discovered that additional information is available to the executive which allows it to perform the failure detection task acceptably. Additional failures might be detected if separate discretes were used for the sync command and for resetting the iteration timing reference. This would allow the local computer to leave its LSC set when it detected itself failed. The other channels would then detect the failure. The local channel's watchdog monitor would not trip out if its iteration timing reference were reset. The failed channel could disengage its actuators and try to resync with the other computers.

Reviewing the information presented in Figure D-6 shows that the selected sync algorithm, Method IV, has all of the desired features. Its operation does not depend on unambiguous failure detection information, which makes the "Master" algorithm unacceptable. It re-evaluates failure status each time, so that the recovery process is made easier. Finally, the memory size and SKEW are the best, considering the functions it performs.

4.4

WATCHDOG MONITOR TRADEOFFS

From the discussion of software frame sync routines in the previous section, it is apparent that the LSC discrete will hold its period and pulse-width within "tens" of microseconds. Considering this, it is reasonable to conclude that the watchdog monitor should have a corresponding time interval tolerance; say, 50 μ sec over a 10 msec frame time. Such precision suggests a digital counter design for the watchdog monitor.

There is, however, a valid tradeoff to be made concerning the use of a less precise, and simpler, analog watchdog monitor design. The tradeoff hinges upon the questions of what constitutes "adequate" second failure coverage, and how significant is the watchdog monitor to this coverage? At this point in the ARCS study only a qualitative answer is possible. The digital design is favored because it offers the "maximum" watchdog monitor performance at a very slight increase in hardware complexity.

4.5

SERVO MONITORING TRADEOFFS

Two primary factors influence the servo monitoring tradeoff; they are:

- Servo-actuator and electronics design
- Reliability requirements.

The central question in the trade-off concerns the location of the boundary between hardware and software servo monitoring. In order to address this question, it is first necessary to establish the failure monitoring requirements in terms of the above two factors.

Figure D-14 presents the baseline ARCS triplex force summed actuation configuration which combines low pressure gain servo valve/actuator modules and cross-channel monitoring with self-monitored, independent channels.

The actuation scheme imposes a requirement that the multiple inputs and feedbacks to the actuator channels track each other within some allowable

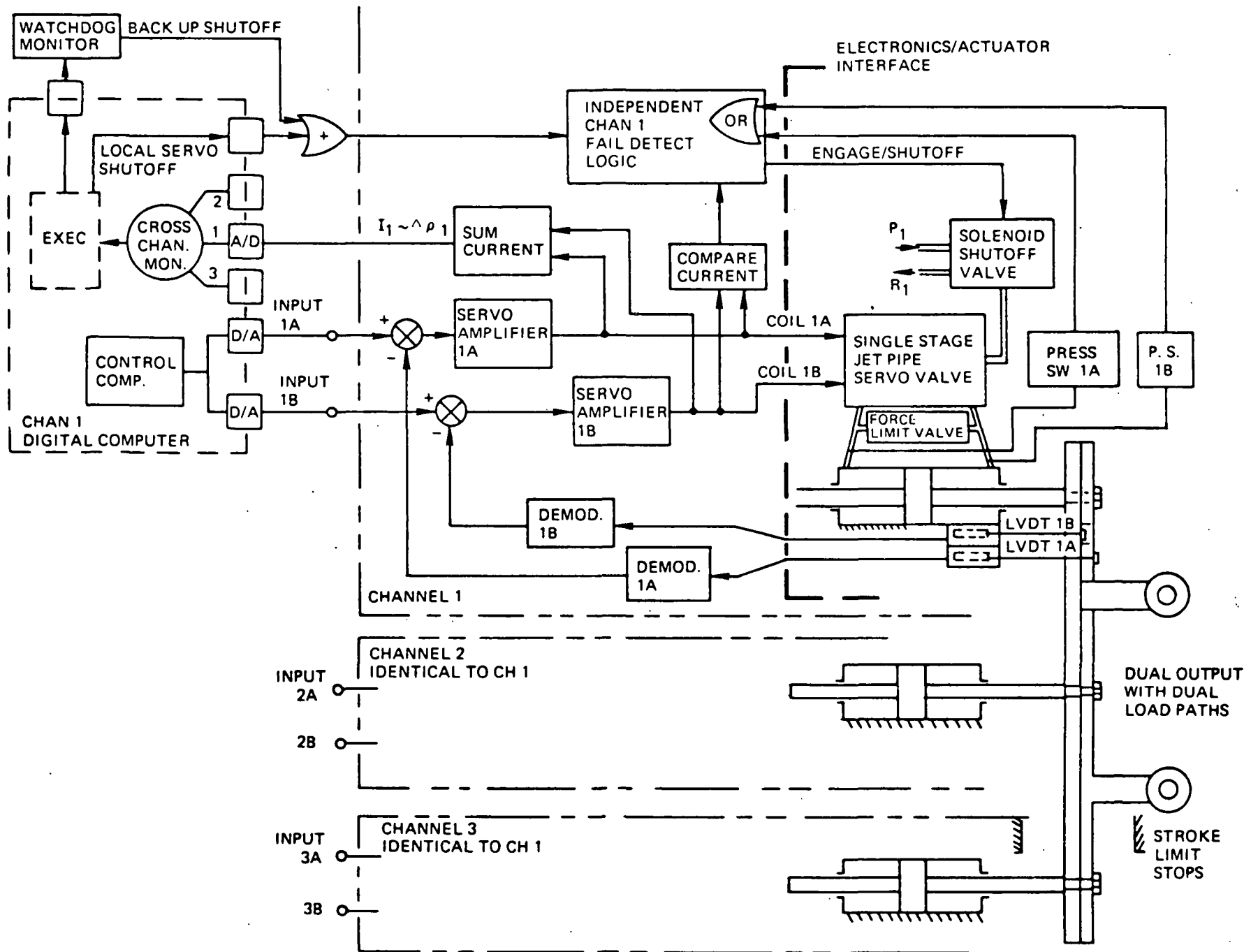


FIGURE D14 CANDIDATE TRIPLEX ACTUATOR FOR ARCS

tolerance. For the actuator feedbacks, this means the LVDT gain tracking as well as the excitation voltage must be controlled within an allowable tolerance. Experience with the 680J actuator and the HLH-ATC actuator has shown that feedback gain tracking can be closely controlled. Multiple analog inputs from the digital/analog converters of synchronous digital computers or from voters in an analog system have been shown to track within tolerances of less than $\pm 1\%$. Previous experience has shown that the basic force summed actuator with single stage, low pressure gain valves, will operate and meet specified performance without the need for differential pressure feedback equalization if the required tracking tolerances are satisfied. Triplex force summed actuators with two stage, high pressure gain valves, such as the HLH-ATC actuators require differential pressure feedback in order to operate without being in a constant, saturated force flight condition.

As shown in Figure D-14, the failure detection/isolation scheme consists of comparison monitoring between dual input and feedback signal paths in each actuator channel as well as cross-channel monitoring of servo valve currents in the digital computer. For a single stage mechanization, the current in the valve is proportional to the differential pressure across the actuator piston.

The independent actuator monitor consists of a current comparator at the output of the two servo amplifiers which can detect a hardover (active) or open (passive) failure in one of the duplicated signal paths. The single comparator will detect feedback failures such as a broken LVDT probe, open LVDT winding and loss of LVDT output due to loss of excitation voltage if each LVDT is excited through separate wires from the power supply in the computer to the LVDT at the actuator. Passive failures in both paths due to loss of electrical power to the servo, amplifiers, demodulators, etc. would have to be protected against by including a power monitor in the fail

detect electronics. The single current monitor would also detect an open wire to the coil of the servo valve or an open in one of the coils itself. If the current driver, short circuit proof type servo amplifier is used to drive the coil of the servo valve and a coil short occurred, the current monitor might not detect this failure. However, an additional comparison of the voltage across the coils could be added if the probability of a short is not considered remote. To protect against hydraulic system pressure loss or a plugged filter or nozzle in the jet pipe servo valve, two pressure switches connected to the extend/retract struts of the actuator are used. Both switch contacts would be closed with loss of pressure and would be used to annunciate the passive failure.

The cross-channel monitor shown as a part of the digital computer provides additional coverage of failure modes not detected by the comparison monitoring. This monitor would detect a failure in the control computation of one computer which feeds both input paths of an actuation channel and could cause a large difference between the currents (pressures) in the three actuation channels. Now the cross-channel monitor could be implemented such that upon second failure the remaining two actuation channels would be shut down. However, with force limit valves contained in each actuator which restrict the force difference between the two remaining actuators to level below the friction of all three channels, the actuator output will not drift hardover upon second failure. Thus, the cross-channel monitor can be inhibited from calling for shutdown on second failure.

The combination of monitoring schemes implemented in the candidate actuation system for ARCS should provide the essentially unit coverage required for first failure isolation in the actuation system. The duplicated signal path comparison monitor provides the required independent, highly reliable output servo monitor required by the ARCS ground rules. The single stage servo valve, triplex force sum actuator and monitoring scheme

4.5

(cont'd)

in Figure D-14 satisfies the ground rule that: "Each computer may control the disconnection of the servo it feeds and shall not directly or in conjunction with other computers effect the operational status of the other servos."

Some other actuation schemes such as the triplex, active/on-line mechanization used on the Boeing-Vertol HLH Direct Electrical Linkage System would not appear to satisfy the ground rule given above since one channel is active and controls the output. If the active channel fails, it must transmit a signal which makes another channel active to control the output.

4.6

NONVOLATILE MEMORY TRADEOFF

Computer systems of the future will be designed to have very low heat dissipation and power usage, and to meet high reliability specifications. For these reasons core memories are being replaced by semiconductor "RAM-ROM" memories, and with this change goes the system's capability of non-volatile "scratchpad" memory.

Presented here are two possible ways to implement a small non-volatile "scratchpad" memory for storage of maintenance information.

C-MOS RAM memories are relatively new in the semiconductor memory market, and have one major advantage over all other semiconductor memories: when not in a cycling mode (just holding data) they use practically zero power supply current. Because of this characteristic, the C-MOS memories can be connected in a circuit, where a battery supplies the necessary power supply current after the normal power supply has been shut down, thereby causing the C-MOS RAM to maintain its storage. Since the leakage current of the battery is larger than the current required by the memory, the battery can supply power to the memory for practically the shelf life of the battery. Rechargeable nickel-cadmium batteries can be used with the battery being charged by the normal power supply during system operation.

Advantages

- Simple low-cost implementation
- Easily interfaced with CPU (fast read and write times)
- Operates from 5V power supply
- Has reliability associated with older technology
- Available from many sources.

Disadvantages

- Nickel-cadmium batteries operate over a limited temperature range (low current requirements of C-MOS will allow operation over a range of up to -40°C to $+65^{\circ}\text{C}$ ambient).
- Batteries must be replaced after a specified time as a maintenance item.
- Memory bits can be lost if power is lost for any amount of time, therefore, maintenance people must be careful, when transporting the memory module, not to short or otherwise effect the battery power to the memory.

MNOS is a new P-channel MOS technology for implementing electrically alterable non-volatile memories (actually electrically alterable ROM's). In these devices binary data is stored in an array of MNOS transistors. The storage mechanism involves a change in threshold of the transistors when charge is injected under the gate electrode by the application of a high voltage (20 to 30 volts).

The technology is very new and therefore has all of the problems associated with a new technology. As a result it can not be seriously considered as a viable solution to the problem at the present time; however, it looks very promising and should be ready by the 1980's.

Advantages

- Data is solidly stored in the memory, maintenance personnel do not have to be careful of the power supply when handling the memory

Disadvantages

- Limited temperature range at present (-25°C to $+70^{\circ}\text{C}$)
- +12V and -12V operating power supply
- -24V and +30V programming power supply
- Memories must periodically be replaced as a maintenance item
- Poor reliability associated with a new technology
- Single-source supply
- Very long write time.

APPENDIX E

FAULT ANALYSIS

The viability of the ARCS reconfiguration process was demonstrated by simulating this process. This appendix discusses three simulation runs that show the system's potential ability to attain normal operation after power on and to continue operation after experiencing transient power faults.

Since power on recovery exercises the major portions of the software needed to recover from a watchdog monitor trip, it was decided that a run with a watchdog monitor would not add to the understanding of that process.

The sensor reconfiguration problem was deemed to be of such magnitude that to simulate it would be far out of the scope of the ARCS study. The power on sequence simulation did show that the interface between computer and SSFD recovery is suitably defined in the ARCS Software Design.

The following paragraphs discuss the simulation results.

Triplex Power-On

Table E-1 shows the hardware indicators for this simulation run. The power for A, B, and C comes on at .02, 0.1, and 0.2 respectively. Power-on interrupts are seen to occur at these times and the iteration reset interrupts begin.

Table E-2 shows the synchronization status of the three computers. A is initially alone as shown by the octal number 4. At 0.1 B comes on and the synchronization is duplex with A and B. At 0.2 C comes on and each computer shows triplex synchronization.

TIME	POWER			WATCH-DOG			POWER INTERRUPT			ITERATION RESET		
	A	B	C	A	B	C	A	B	C	A	B	C
.0000	F	F	F	F	F	F	F	F	F	F	F	F
.0200	T	F	F	T	F	F	T	F	F	T	F	F
.0400	T	F	F	T	F	F	F	F	F	T	F	F
.0600	T	F	F	T	F	F	F	F	F	T	F	F
.0800	T	F	F	T	F	F	F	F	F	T	F	F
.1000	T	T	F	T	T	F	F	T	F	T	T	F
.1200	T	T	F	T	T	F	F	F	F	T	T	F
.1400	T	T	F	T	T	F	F	F	F	T	T	F
.1600	T	T	F	T	T	F	F	F	F	T	T	F
.1800	T	T	F	T	T	F	F	F	F	T	T	F
.2000	T	T	T	T	T	T	F	F	T	T	T	T
.2200	T	T	T	T	T	T	F	F	F	T	T	T
.2400	T	T	T	T	T	T	F	F	F	T	T	T
.2600	T	T	T	T	T	T	F	F	F	T	T	T
.2800	T	T	T	T	T	T	F	F	F	T	T	T
.3000	T	T	T	T	T	T	F	F	F	T	T	T
.3200	T	T	T	T	T	T	F	F	F	T	T	T
.3400	T	T	T	T	T	T	F	F	F	T	T	T
.3600	T	T	T	T	T	T	F	F	F	T	T	T
.3800	T	T	T	T	T	T	F	F	F	T	T	T
.4000	T	T	T	T	T	T	F	F	F	T	T	T
.4200	T	T	T	T	T	T	F	F	F	T	T	T
.4400	T	T	T	T	T	T	F	F	F	T	T	T
.4600	T	T	T	T	T	T	F	F	F	T	T	T
.4800	T	T	T	T	T	T	F	F	F	T	T	T
.5000	T	T	T	T	T	T	F	F	F	T	T	T
.5200	T	T	T	T	T	T	F	F	F	T	T	T
.5400	T	T	T	T	T	T	F	F	F	T	T	T
.5600	T	T	T	T	T	T	F	F	F	T	T	T
.5800	T	T	T	T	T	T	F	F	F	T	T	T
.6000	T	T	T	T	T	T	F	F	F	T	T	T
.6200	T	T	T	T	T	T	F	F	F	T	T	T
.6400	T	T	T	T	T	T	F	F	F	T	T	T
.6600	T	T	T	T	T	T	F	F	F	T	T	T
.6800	T	T	T	T	T	T	F	F	F	T	T	T
.7000	T	T	T	T	T	T	F	F	F	T	T	T
.7200	T	T	T	T	T	T	F	F	F	T	T	T
.7400	T	T	T	T	T	T	F	F	F	T	T	T
.7600	T	T	T	T	T	T	F	F	F	T	T	T
.7800	T	T	T	T	T	T	F	F	F	T	T	T
.8000	T	T	T	T	T	T	F	F	F	T	T	T
.8200	T	T	T	T	T	T	F	F	F	T	T	T
.8400	T	T	T	T	T	T	F	F	F	T	T	T
.8600	T	T	T	T	T	T	F	F	F	T	T	T
.8800	T	T	T	T	T	T	F	F	F	T	T	T
.9000	T	T	T	T	T	T	F	F	F	T	T	T
.9200	T	T	T	T	T	T	F	F	F	T	T	T
.9400	T	T	T	T	T	T	F	F	F	T	T	T
.9600	T	T	T	T	T	T	F	F	F	T	T	T
.9800	T	T	T	T	T	T	F	F	F	T	T	T
1.0000	T	T	T	T	T	T	F	F	F	T	T	T

TABLE: E-1 SIMULATED HARDWARE INDICATORS

TIME	A	B	C
.000	0	0	0
.020	4	0	0
.040	4	0	0
.060	4	0	0
.080	4	0	0
.100	6	6	0
.120	6	6	0
.140	6	6	0
.160	6	6	0
.180	6	6	0
.200	7	7	7
.220	7	7	7
.240	7	7	7
.260	7	7	7
.280	7	7	7
.300	7	7	7
.320	7	7	7
.340	7	7	7
.360	7	7	7
.380	7	7	7
.400	7	7	7
.420	7	7	7
.440	7	7	7
.460	7	7	7
.480	7	7	7
.500	7	7	7
.520	7	7	7
.540	7	7	7
.560	7	7	7
.580	7	7	7
.600	7	7	7
.620	7	7	7
.640	7	7	7
.660	7	7	7
.680	7	7	7
.700	7	7	7
.720	7	7	7
.740	7	7	7
.760	7	7	7
.780	7	7	7
.800	7	7	7
.820	7	7	7
.840	7	7	7
.860	7	7	7
.880	7	7	7
.900	7	7	7
.920	7	7	7
.940	7	7	7
.960	7	7	7
.980	7	7	7
1.000	7	7	7

TABLE: E-2 SYNCHRONIZATION INDICATORS

ORIGINAL PAGE IS
OF POOR QUALITY

Tables E-3, E-4, and E-5 show the recovery indicators for the A, B, and C computers. At 0.02 A starts power-on recovery in simplex. At 0.1 B comes on A sets B recovery request flag and immediately releases its B permanents since no other computer is attempting recovery. At the same time B releases its own permanents and starts recovery.

At 0.2 C comes on. Since B is still recovering C's permanents are not released. This can be seen in Tables E-6, E-7, and E-8. At 0.22 B's do-not-use is reset, therefore C can begin recovery at 0.24. Until this time, C had run as if it were B.

Tables E-9, E-10, and E-11 show the input from the sensor, the selected signal, the limited input to the filter, and the filter output. The point to note is that the recovery does not introduce any errors in processing.

351

TIME		RECOVERY A		RECOVERY REQUEST	
		A	B	C	
.000	-1	0	0	0	
.020	1	0	0	0	
.040	0	0	0	0	
.060	0	0	0	0	
.080	0	0	0	0	
.100	4	0	0	0	
.120	0	0	0	0	
.140	0	0	0	0	
.160	0	0	0	0	
.180	0	0	0	0	
.200	3	0	0	1	
.220	0	0	0	1	
.240	0	0	0	0	
.260	0	0	0	0	
.280	0	0	0	0	
.300	0	0	0	0	
.320	0	0	0	0	
.340	0	0	0	0	
.360	0	0	0	0	
.380	0	0	0	0	
.400	0	0	0	0	
.420	0	0	0	0	
.440	0	0	0	0	
.460	0	0	0	0	
.480	0	0	0	0	
.500	0	0	0	0	
.520	0	0	0	0	
.540	0	0	0	0	
.560	0	0	0	0	
.580	0	0	0	0	
.600	0	0	0	0	
.620	0	0	0	0	
.640	0	0	0	0	
.660	0	0	0	0	
.680	0	0	0	0	
.700	0	0	0	0	
.720	0	0	0	0	
.740	0	0	0	0	
.760	0	0	0	0	
.780	0	0	0	0	
.800	0	0	0	0	
.820	0	0	0	0	
.840	0	0	0	0	
.860	0	0	0	0	
.880	0	0	0	0	
.900	0	0	0	0	
.920	0	0	0	0	
.940	0	0	0	0	
.960	0	0	0	0	
.980	0	0	0	0	
1.000	0	0	0	0	

TABLE: E-3 RECOVERY INDICATORS FOR COMPUTER A

TIME		RECOVERY B		
		A	B	C
.000	-1	0	0	0
.020	-1	0	0	0
.040	-1	0	0	0
.060	-1	0	0	0
.080	-1	0	0	0
.100	1	0	0	0
.120	0	0	0	0
.140	0	0	0	0
.160	0	0	0	0
.180	0	0	0	0
.200	4	0	0	1
.220	0	0	0	1
.240	0	0	0	0
.260	0	0	0	0
.280	0	0	0	0
.300	0	0	0	0
.320	0	0	0	0
.340	0	0	0	0
.360	0	0	0	0
.380	0	0	0	0
.400	0	0	0	0
.420	0	0	0	0
.440	0	0	0	0
.460	0	0	0	0
.480	0	0	0	0
.500	0	0	0	0
.520	0	0	0	0
.540	0	0	0	0
.560	0	0	0	0
.580	0	0	0	0
.600	0	0	0	0
.620	0	0	0	0
.640	0	0	0	0
.660	0	0	0	0
.680	0	0	0	0
.700	0	0	0	0
.720	0	0	0	0
.740	0	0	0	0
.760	0	0	0	0
.780	0	0	0	0
.800	0	0	0	0
.820	0	0	0	0
.840	0	0	0	0
.860	0	0	0	0
.880	0	0	0	0
.900	0	0	0	0
.920	0	0	0	0
.940	0	0	0	0
.960	0	0	0	0
.980	0	0	0	0
1.000	0	0	0	0

TABLE: E-4 RECOVERY INDICATORS FOR COMPUTER B

TIME		RECOVERY C			RECOVERY REQUEST		
		A	B	C	A	B	C
.000	-1	0	0	0	0	0	0
.020	-1	0	0	0	0	0	0
.040	-1	0	0	0	0	0	0
.060	-1	0	0	0	0	0	0
.080	-1	0	0	0	0	0	0
.100	-1	0	0	0	0	0	0
.120	-1	0	0	0	0	0	0
.140	-1	0	0	0	0	0	0
.160	-1	0	0	0	0	0	0
.180	-1	0	0	0	0	0	0
.200	1	0	0	1	0	0	1
.220	0	0	0	1	0	0	1
.240	0	0	0	0	0	0	0
.260	0	0	0	0	0	0	0
.280	0	0	0	0	0	0	0
.300	0	0	0	0	0	0	0
.320	0	0	0	0	0	0	0
.340	0	0	0	0	0	0	0
.360	0	0	0	0	0	0	0
.380	0	0	0	0	0	0	0
.400	0	0	0	0	0	0	0
.420	0	0	0	0	0	0	0
.440	0	0	0	0	0	0	0
.460	0	0	0	0	0	0	0
.480	0	0	0	0	0	0	0
.500	0	0	0	0	0	0	0
.520	0	0	0	0	0	0	0
.540	0	0	0	0	0	0	0
.560	0	0	0	0	0	0	0
.580	0	0	0	0	0	0	0
.600	0	0	0	0	0	0	0
.620	0	0	0	0	0	0	0
.640	0	0	0	0	0	0	0
.660	0	0	0	0	0	0	0
.680	0	0	0	0	0	0	0
.700	0	0	0	0	0	0	0
.720	0	0	0	0	0	0	0
.740	0	0	0	0	0	0	0
.760	0	0	0	0	0	0	0
.780	0	0	0	0	0	0	0
.800	0	0	0	0	0	0	0
.820	0	0	0	0	0	0	0
.840	0	0	0	0	0	0	0
.860	0	0	0	0	0	0	0
.880	0	0	0	0	0	0	0
.900	0	0	0	0	0	0	0
.920	0	0	0	0	0	0	0
.940	0	0	0	0	0	0	0
.960	0	0	0	0	0	0	0
.980	0	0	0	0	0	0	0
1.000	0	0	0	0	0	0	0

TABLE: E-5 RECOVERY INDICATORS FOR COMPUTER C.

TIME	DO NOT USE			PERMANENT			T1 COUNT			T2 COUNT		
	A	B	C	A	B	C	A	B	C	A	B	C
.000	0	0	0	0	0	0	0	0	0	0	0	0
.020	0	0	0	0	1	1	0	0	0	0	0	0
.040	0	0	0	0	1	1	0	0	0	0	0	0
.060	0	0	0	0	1	1	0	0	0	0	0	0
.080	0	0	0	0	1	1	0	0	0	0	0	0
.100	0	1	0	0	0	1	0	1	0	0	1	0
.120	0	1	0	0	0	1	0	2	0	0	2	0
.140	0	1	0	0	0	1	0	3	0	0	3	0
.160	0	1	0	0	0	1	0	4	0	0	4	0
.180	0	1	0	0	0	1	0	5	0	0	5	0
.200	0	1	0	0	0	1	0	6	0	0	6	0
.220	0	0	0	0	0	1	0	0	0	0	0	0
.240	0	0	1	0	0	0	0	0	1	0	0	1
.260	0	0	1	0	0	0	0	0	2	0	0	2
.280	0	0	1	0	0	0	0	0	3	0	0	3
.300	0	0	1	0	0	0	0	0	4	0	0	4
.320	0	0	1	0	0	0	0	0	5	0	0	5
.340	0	0	1	0	0	0	0	0	6	0	0	6
.360	0	0	0	0	0	0	0	0	0	0	0	0
.380	0	0	0	0	0	0	0	0	0	0	0	0
.400	0	0	0	0	0	0	0	0	0	0	0	0
.420	0	0	0	0	0	0	0	0	0	0	0	0
.440	0	0	0	0	0	0	0	0	0	0	0	0
.460	0	0	0	0	0	0	0	0	0	0	0	0
.480	0	0	0	0	0	0	0	0	0	0	0	0
.500	0	0	0	0	0	0	0	0	0	0	0	0
.520	0	0	0	0	0	0	0	0	0	0	0	0
.540	0	0	0	0	0	0	0	0	0	0	0	0
.560	0	0	0	0	0	0	0	0	0	0	0	0
.580	0	0	0	0	0	0	0	0	0	0	0	0
.600	0	0	0	0	0	0	0	0	0	0	0	0
.620	0	0	0	0	0	0	0	0	0	0	0	0
.640	0	0	0	0	0	0	0	0	0	0	0	0
.660	0	0	0	0	0	0	0	0	0	0	0	0
.680	0	0	0	0	0	0	0	0	0	0	0	0
.700	0	0	0	0	0	0	0	0	0	0	0	0
.720	0	0	0	0	0	0	0	0	0	0	0	0
.740	0	0	0	0	0	0	0	0	0	0	0	0
.760	0	0	0	0	0	0	0	0	0	0	0	0
.780	0	0	0	0	0	0	0	0	0	0	0	0
.800	0	0	0	0	0	0	0	0	0	0	0	0
.820	0	0	0	0	0	0	0	0	0	0	0	0
.840	0	0	0	0	0	0	0	0	0	0	0	0
.860	0	0	0	0	0	0	0	0	0	0	0	0
.880	0	0	0	0	0	0	0	0	0	0	0	0
.900	0	0	0	0	0	0	0	0	0	0	0	0
.920	0	0	0	0	0	0	0	0	0	0	0	0
.940	0	0	0	0	0	0	0	0	0	0	0	0
.960	0	0	0	0	0	0	0	0	0	0	0	0
.980	0	0	0	0	0	0	0	0	0	0	0	0
1.000	0	0	0	0	0	0	0	0	0	0	0	0

TABLE: E-6 DO NOT USE, PERMANENT FLAGS AND COUNTERS
FOR A, B, AND C SIGNALS FOR COMPUTER A

TIME	DO NOT USE			PERMANENT			T1 COUNT			T2 COUNT		
	A	B	C	A	B	C	A	B	C	A	B	C
.000	0	0	0	0	0	0	0	0	0	0	0	0
.020	0	0	0	0	0	0	0	0	0	0	0	0
.040	0	0	0	0	0	0	0	0	0	0	0	0
.060	0	0	0	0	0	0	0	0	0	0	0	0
.080	0	0	0	0	0	0	0	0	0	0	0	0
.100	0	1	0	0	0	1	0	1	0	0	1	0
.120	0	1	0	0	0	1	0	2	0	0	2	0
.140	0	1	0	0	0	1	0	3	0	0	3	0
.160	0	1	0	0	0	1	0	4	0	0	4	0
.180	0	1	0	0	0	1	0	5	0	0	5	0
.200	0	1	0	0	0	1	0	6	0	0	6	0
.220	0	0	0	0	0	1	0	7	0	0	7	0
.240	0	0	1	0	0	0	0	0	1	0	0	1
.260	0	0	1	0	0	0	0	0	2	0	0	2
.280	0	0	1	0	0	0	0	0	3	0	0	3
.300	0	0	1	0	0	0	0	0	4	0	0	4
.320	0	0	1	0	0	0	0	0	5	0	0	5
.340	0	0	1	0	0	0	0	0	6	0	0	6
.360	0	0	0	0	0	0	0	0	0	0	0	0
.380	0	0	0	0	0	0	0	0	0	0	0	0
.400	0	0	0	0	0	0	0	0	0	0	0	0
.420	0	0	0	0	0	0	0	0	0	0	0	0
.440	0	0	0	0	0	0	0	0	0	0	0	0
.460	0	0	0	0	0	0	0	0	0	0	0	0
.480	0	0	0	0	0	0	0	0	0	0	0	0
.500	0	0	0	0	0	0	0	0	0	0	0	0
.520	0	0	0	0	0	0	0	0	0	0	0	0
.540	0	0	0	0	0	0	0	0	0	0	0	0
.560	0	0	0	0	0	0	0	0	0	0	0	0
.580	0	0	0	0	0	0	0	0	0	0	0	0
.600	0	0	0	0	0	0	0	0	0	0	0	0
.620	0	0	0	0	0	0	0	0	0	0	0	0
.640	0	0	0	0	0	0	0	0	0	0	0	0
.660	0	0	0	0	0	0	0	0	0	0	0	0
.680	0	0	0	0	0	0	0	0	0	0	0	0
.700	0	0	0	0	0	0	0	0	0	0	0	0
.720	0	0	0	0	0	0	0	0	0	0	0	0
.740	0	0	0	0	0	0	0	0	0	0	0	0
.760	0	0	0	0	0	0	0	0	0	0	0	0
.780	0	0	0	0	0	0	0	0	0	0	0	0
.800	0	0	0	0	0	0	0	0	0	0	0	0
.820	0	0	0	0	0	0	0	0	0	0	0	0
.840	0	0	0	0	0	0	0	0	0	0	0	0
.860	0	0	0	0	0	0	0	0	0	0	0	0
.880	0	0	0	0	0	0	0	0	0	0	0	0
.900	0	0	0	0	0	0	0	0	0	0	0	0
.920	0	0	0	0	0	0	0	0	0	0	0	0
.940	0	0	0	0	0	0	0	0	0	0	0	0
.960	0	0	0	0	0	0	0	0	0	0	0	0
.980	0	0	0	0	0	0	0	0	0	0	0	0
1.000	0	0	0	0	0	0	0	0	0	0	0	0

TABLE E-7 DO NOT USE PERMANENT FLAGS AND COUNTERS FOR A, B, AND C SIGNALS FOR COMPUTER B

DO NOT USE
PERMANENT

TIME							T1 COUNT			T2 COUNT		
	A	B	C	A	B	C	A	B	C	A	B	C
.000	0	0	0	0	0	0	0	0	0	0	0	0
.020	0	0	0	0	0	0	0	0	0	0	0	0
.040	0	0	0	0	0	0	0	0	0	0	0	0
.060	0	0	0	0	0	0	0	0	0	0	0	0
.080	0	0	0	0	0	0	0	0	0	0	0	0
.100	0	0	0	0	0	0	0	0	0	0	0	0
.120	0	0	0	0	0	0	0	0	0	0	0	0
.140	0	0	0	0	0	0	0	0	0	0	0	0
.160	0	0	0	0	0	0	0	0	0	0	0	0
.180	0	0	0	0	0	0	0	0	0	0	0	0
.200	0	1	0	0	0	1	0	6	0	0	0	0
.220	0	0	0	0	0	1	0	0	0	0	0	0
.240	0	0	1	0	0	0	0	0	1	0	0	1
.260	0	0	1	0	0	0	0	0	2	0	0	2
.280	0	0	1	0	0	0	0	0	3	0	0	3
.300	0	0	1	0	0	0	0	0	4	0	0	4
.320	0	0	1	0	0	0	0	0	5	0	0	5
.340	0	0	1	0	0	0	0	0	6	0	0	6
.360	0	0	0	0	0	0	0	0	0	0	0	0
.380	0	0	0	0	0	0	0	0	0	0	0	0
.400	0	0	0	0	0	0	0	0	0	0	0	0
.420	0	0	0	0	0	0	0	0	0	0	0	0
.440	0	0	0	0	0	0	0	0	0	0	0	0
.460	0	0	0	0	0	0	0	0	0	0	0	0
.480	0	0	0	0	0	0	0	0	0	0	0	0
.500	0	0	0	0	0	0	0	0	0	0	0	0
.520	0	0	0	0	0	0	0	0	0	0	0	0
.540	0	0	0	0	0	0	0	0	0	0	0	0
.560	0	0	0	0	0	0	0	0	0	0	0	0
.580	0	0	0	0	0	0	0	0	0	0	0	0
.600	0	0	0	0	0	0	0	0	0	0	0	0
.620	0	0	0	0	0	0	0	0	0	0	0	0
.640	0	0	0	0	0	0	0	0	0	0	0	0
.660	0	0	0	0	0	0	0	0	0	0	0	0
.680	0	0	0	0	0	0	0	0	0	0	0	0
.700	0	0	0	0	0	0	0	0	0	0	0	0
.720	0	0	0	0	0	0	0	0	0	0	0	0
.740	0	0	0	0	0	0	0	0	0	0	0	0
.760	0	0	0	0	0	0	0	0	0	0	0	0
.780	0	0	0	0	0	0	0	0	0	0	0	0
.800	0	0	0	0	0	0	0	0	0	0	0	0
.820	0	0	0	0	0	0	0	0	0	0	0	0
.840	0	0	0	0	0	0	0	0	0	0	0	0
.860	0	0	0	0	0	0	0	0	0	0	0	0
.880	0	0	0	0	0	0	0	0	0	0	0	0
.900	0	0	0	0	0	0	0	0	0	0	0	0
.920	0	0	0	0	0	0	0	0	0	0	0	0
.940	0	0	0	0	0	0	0	0	0	0	0	0
.960	0	0	0	0	0	0	0	0	0	0	0	0
.980	0	0	0	0	0	0	0	0	0	0	0	0
1.000	0	0	0	0	0	0	0	0	0	0	0	0

TABLE: E-8 DO NOT USE, PERMANENT FLAGS AND COUNTERS
FOR A, B, AND C SIGNALS FOR COMPUTER C

TIME	SENSOR A	SELECTED SIGNAL	INPUT TO FILTER	OUTPUT OF FILTER
.000	.0000	.0000	.0000	.0000
.020	.7431	.7431	.7431	.2705
.040	.9945	.9945	.9945	.8303
.060	.5878	.5878	.5878	.9355
.080	-.2079	-.2079	-.2079	.3075
.100	-.8660	-.8660	-.8660	-.5115
.120	-.9511	-.9511	-.9511	-.9738
.140	-.4067	-.4067	-.4067	-.7995
.160	.4067	.4067	.4067	-.0909
.180	.9511	.9511	.9511	.6711
.200	.8660	.8660	.8660	.9948
.220	.2079	.2079	.2079	.6599
.240	-.5878	-.5878	-.5878	-.1115
.260	-.9945	-.9945	-.9945	-.8091
.280	-.7431	-.7431	-.7431	-.9713
.300	-.0000	-.0000	-.0000	-.4908
.320	.7431	.7431	.7431	.3146
.340	.9945	.9945	.9945	.9117
.360	.5878	.5878	.5878	.9056
.380	-.2079	-.2079	-.2079	.3001
.400	-.8660	-.8660	-.8660	-.5039
.420	-.9511	-.9511	-.9511	-.9745
.440	-.4067	-.4067	-.4067	-.8002
.460	.4067	.4067	.4067	-.0964
.480	.9511	.9511	.9511	.6712
.500	.8660	.8660	.8660	.9946
.520	.2079	.2079	.2079	.6599
.540	-.5878	-.5878	-.5878	-.1115
.560	-.9945	-.9945	-.9945	-.8091
.580	-.7431	-.7431	-.7431	-.9713
.600	-.0000	-.0000	-.0000	-.4908
.620	.7431	.7431	.7431	.3146
.640	.9945	.9945	.9945	.9117
.660	.5878	.5878	.5878	.9056
.680	-.2079	-.2079	-.2079	.3001
.700	-.8660	-.8660	-.8660	-.5039
.720	-.9511	-.9511	-.9511	-.9745
.740	-.4067	-.4067	-.4067	-.8002
.760	.4067	.4067	.4067	-.0964
.780	.9511	.9511	.9511	.6712
.800	.8660	.8660	.8660	.9946
.820	.2079	.2079	.2079	.6599
.840	-.5878	-.5878	-.5878	-.1115
.860	-.9945	-.9945	-.9945	-.8091
.880	-.7431	-.7431	-.7431	-.9713
.900	-.0000	-.0000	-.0000	-.4908
.920	.7431	.7431	.7431	.3146
.940	.9945	.9945	.9945	.9117
.960	.5878	.5878	.5878	.9056
.980	-.2079	-.2079	-.2079	.3001
1.000	-.8660	-.8660	-.8660	-.5039

TIME	SENSOR B	SELECTED SIGNAL	INPUT TO FILTER	OUTPUT OF FILTERS
.000	.0000	.0000	.0000	.0000
.020	.0000	.0000	.0000	.0000
.040	.0000	.0000	.0000	.0000
.060	.0000	.0000	.0000	.0000
.080	.0000	.0000	.0000	.0000
.100	-.8660	-.8660	-.8660	-.5115
.120	-.9511	-.9511	-.9511	-.9738
.140	-.4067	-.4067	-.4067	-.7990
.160	.4067	.4067	.4067	-.0969
.180	.9511	.9511	.9511	.6711
.200	.8660	.8660	.8660	.9948
.220	.2079	.2079	.2079	.6599
.240	-.5878	-.5878	-.5878	-.1115
.260	-.9945	-.9945	-.9945	-.8091
.280	-.7431	-.7431	-.7431	-.9713
.300	-.0000	-.0000	-.0000	-.4908
.320	.7431	.7431	.7431	.3146
.340	.9945	.9945	.9945	.9117
.360	.5878	.5878	.5878	.9056
.380	-.2079	-.2079	-.2079	.3001
.400	-.8660	-.8660	-.8660	-.5039
.420	-.9511	-.9511	-.9511	-.9745
.440	-.4067	-.4067	-.4067	-.8002
.460	.4067	.4067	.4067	-.0964
.480	.9511	.9511	.9511	.6712
.500	.8660	.8660	.8660	.9946
.520	.2079	.2079	.2079	.6599
.540	-.5878	-.5878	-.5878	-.1115
.560	-.9945	-.9945	-.9945	-.8091
.580	-.7431	-.7431	-.7431	-.9713
.600	-.0000	-.0000	-.0000	-.4908
.620	.7431	.7431	.7431	.3146
.640	.9945	.9945	.9945	.9117
.660	.5878	.5878	.5878	.9056
.680	-.2079	-.2079	-.2079	.3001
.700	-.8660	-.8660	-.8660	-.5039
.720	-.9511	-.9511	-.9511	-.9745
.740	-.4067	-.4067	-.4067	-.8002
.760	.4067	.4067	.4067	-.0964
.780	.9511	.9511	.9511	.6712
.800	.8660	.8660	.8660	.9946
.820	.2079	.2079	.2079	.6599
.840	-.5878	-.5878	-.5878	-.1115
.860	-.9945	-.9945	-.9945	-.8091
.880	-.7431	-.7431	-.7431	-.9713
.900	-.0000	-.0000	-.0000	-.4908
.920	.7431	.7431	.7431	.3146
.940	.9945	.9945	.9945	.9117
.960	.5878	.5878	.5878	.9056
.980	-.2079	-.2079	-.2079	.3001
1.000	-.8660	-.8660	-.8660	-.5039

TABLE E-10 INPUT AND OUTPUT FOR COMPUTER B

TIME	SENSOR C	SELECTED SIGNAL	INPUT TO FILTER	OUTPUT OF FILTER
.000	.0000	.0000	.0000	.0000
.020	.0000	.0000	.0000	.0000
.040	.0000	.0000	.0000	.0000
.060	.0000	.0000	.0000	.0000
.080	.0000	.0000	.0000	.0000
.100	.0000	.0000	.0000	.0000
.120	.0000	.0000	.0000	.0000
.140	.0000	.0000	.0000	.0000
.160	.0000	.0000	.0000	.0000
.180	.0000	.0000	.0000	.0000
.200	.8660	.8660	.8660	.9948
.220	.2079	.2079	.2079	.6599
.240	-.5878	-.5878	-.5878	-.1115
.260	-.9945	-.9945	-.9945	-.8091
.280	-.7431	-.7431	-.7431	-.9713
.300	-.0000	-.0000	-.0000	-.4908
.320	.7431	.7431	.7431	.3146
.340	.9945	.9945	.9945	.9117
.360	.5878	.5878	.5878	.9056
.380	-.2079	-.2079	-.2079	.3001
.400	-.8660	-.8660	-.8660	-.5039
.420	-.9511	-.9511	-.9511	-.9745
.440	-.4067	-.4067	-.4067	-.8002
.460	.4067	.4067	.4067	-.0964
.480	.9511	.9511	.9511	.6712
.500	.8660	.8660	.8660	.9946
.520	.2079	.2079	.2079	.6599
.540	-.5878	-.5878	-.5878	-.1115
.560	-.9945	-.9945	-.9945	-.8091
.580	-.7431	-.7431	-.7431	-.9713
.600	-.0000	-.0000	-.0000	-.4908
.620	.7431	.7431	.7431	.3146
.640	.9945	.9945	.9945	.9117
.660	.5878	.5878	.5878	.9056
.680	-.2079	-.2079	-.2079	.3001
.700	-.8660	-.8660	-.8660	-.5039
.720	-.9511	-.9511	-.9511	-.9745
.740	-.4067	-.4067	-.4067	-.8002
.760	.4067	.4067	.4067	-.0964
.780	.9511	.9511	.9511	.6712
.800	.8660	.8660	.8660	.9946
.820	.2079	.2079	.2079	.6599
.840	-.5878	-.5878	-.5878	-.1115
.860	-.9945	-.9945	-.9945	-.8091
.880	-.7431	-.7431	-.7431	-.9713
.900	-.0000	-.0000	-.0000	-.4908
.920	.7431	.7431	.7431	.3146
.940	.9945	.9945	.9945	.9117
.960	.5878	.5878	.5878	.9056
.980	-.2079	-.2079	-.2079	.3001
1.000	-.8660	-.8660	-.8660	-.5039

TABLE E-11 INPUT AND OUTPUT FOR COMPUTER C

Single Power Fault

Table E-12 shows that the A computer experienced a power fault from 0.40 to 0.58 that caused power loss. At 0.60 power to A comes back on and recovery proceeds.

Table E-13 indicates that as expected the A computer does not synchronize and the system goes duplex with B and C.

Tables E-14 and E-15 indicate that at 0.60 the A computer needs recovery based on the fact that it has just synchronized with the others. Table E-16 indicates A requiring power-on recovery at 0.60.

Tables E-17, E-18, and E-19 show that at 0.60 A's permanent flag is released and its do-not-use flag is set. After a time T1 A is again accepted as good and the system is again triplex.

Tables E-20, E-21, and E-22 show the inputs and outputs. Again note that the recovery does not produce any problems.

TIME	POWER			WATCH-DOG			POWER INTERUPT			ITERATION RESET		
	A	B	C	A	B	C	A	B	C	A	B	C
.0000	F	F	F	F	F	F	F	F	F	F	F	F
.0200	T	F	F	T	F	F	T	F	F	T	F	F
.0400	T	F	F	T	F	F	F	F	F	T	F	F
.0600	T	F	F	T	F	F	F	F	F	T	F	F
.0800	T	F	F	T	F	F	F	F	F	T	F	F
.1000	T	T	F	T	T	F	F	T	F	T	T	F
.1200	T	T	F	T	T	F	F	F	F	T	T	F
.1400	T	T	F	T	T	F	F	F	F	T	T	F
.1600	T	T	F	T	T	F	F	F	F	T	T	F
.1800	T	T	F	T	T	F	F	F	F	T	T	F
.2000	T	T	T	T	T	T	F	F	T	T	T	T
.2200	T	T	T	T	T	T	F	F	F	T	T	T
.2400	T	T	T	T	T	T	F	F	F	T	T	T
.2600	T	T	T	T	T	T	F	F	F	T	T	T
.2800	T	T	T	T	T	T	F	F	F	T	T	T
.3000	T	T	T	T	T	T	F	F	F	T	T	T
.3200	T	T	T	T	T	T	F	F	F	T	T	T
.3400	T	T	T	T	T	T	F	F	F	T	T	T
.3600	T	T	T	T	T	T	F	F	F	T	T	T
.3800	T	T	T	T	T	T	F	F	F	T	T	T
.4000	F	T	T	F	T	T	F	F	F	T	T	T
.4200	F	T	T	F	T	T	F	F	F	F	T	T
.4400	F	T	T	F	T	T	F	F	F	F	T	T
.4600	F	T	T	F	T	T	F	F	F	F	T	T
.4800	F	T	T	F	T	T	F	F	F	F	T	T
.5000	F	T	T	F	T	T	F	F	F	F	T	T
.5200	F	T	T	F	T	T	F	F	F	F	T	T
.5400	F	T	T	F	T	T	F	F	F	F	T	T
.5600	F	T	T	F	T	T	F	F	F	F	T	T
.5800	F	T	T	F	T	T	F	F	F	F	T	T
.6000	T	T	T	T	T	T	T	F	F	T	T	T
.6200	T	T	T	T	T	T	F	F	F	T	T	T
.6400	T	T	T	T	T	T	F	F	F	T	T	T
.6600	T	T	T	T	T	T	F	F	F	T	T	T
.6800	T	T	T	T	T	T	F	F	F	T	T	T
.7000	T	T	T	T	T	T	F	F	F	T	T	T
.7200	T	T	T	T	T	T	F	F	F	T	T	T
.7400	T	T	T	T	T	T	F	F	F	T	T	T
.7600	T	T	T	T	T	T	F	F	F	T	T	T
.7800	T	T	T	T	T	T	F	F	F	T	T	T
.8000	T	T	T	T	T	T	F	F	F	T	T	T
.8200	T	T	T	T	T	T	F	F	F	T	T	T
.8400	T	T	T	T	T	T	F	F	F	T	T	T
.8600	T	T	T	T	T	T	F	F	F	T	T	T
.8800	T	T	T	T	T	T	F	F	F	T	T	T
.9000	T	T	T	T	T	T	F	F	F	T	T	T
.9200	T	T	T	T	T	T	F	F	F	T	T	T
.9400	T	T	T	T	T	T	F	F	F	T	T	T
.9600	T	T	T	T	T	T	F	F	F	T	T	T
.9800	T	T	T	T	T	T	F	F	F	T	T	T
1.0000	T	T	T	T	T	T	F	F	F	T	T	T

TABLE:E-12 SIMULATED HARDWARE INDICATORS

TIME	A	B	C
.000	0	0	0
.020	4	0	0
.040	4	0	0
.060	4	0	0
.080	4	0	0
.100	6	6	0
.120	6	6	0
.140	6	6	0
.160	6	6	0
.180	6	6	0
.200	7	7	7
.220	7	7	7
.240	7	7	7
.260	7	7	7
.280	7	7	7
.300	7	7	7
.320	7	7	7
.340	7	7	7
.360	7	7	7
.380	7	7	7
.400	0	3	3
.420	0	3	3
.440	0	3	3
.460	0	3	3
.480	0	3	3
.500	0	3	3
.520	0	3	3
.540	0	3	3
.560	0	3	3
.580	0	3	3
.600	7	7	7
.620	7	7	7
.640	7	7	7
.660	7	7	7
.680	7	7	7
.700	7	7	7
.720	7	7	7
.740	7	7	7
.760	7	7	7
.780	7	7	7
.800	7	7	7
.820	7	7	7
.840	7	7	7
.860	7	7	7
.880	7	7	7
.900	7	7	7
.920	7	7	7
.940	7	7	7
.960	7	7	7
.980	7	7	7
1.000	7	7	7

TABLE E-13 SYNCHRONIZATION INDICATORS.

TIME		RECOVERY REQUEST		
		A	B	C
.000	-1	0	0	0
.020	1	0	0	0
.040	0	0	0	0
.060	0	0	0	0
.080	0	0	0	0
.100	0	0	0	0
.120	0	0	0	0
.140	0	0	0	0
.160	0	0	0	0
.180	0	0	0	0
.200	3	0	0	1
.220	0	0	0	1
.240	0	0	0	0
.260	0	0	0	0
.280	0	0	0	0
.300	0	0	0	0
.320	0	0	0	0
.340	0	0	0	0
.360	0	0	0	0
.380	0	0	0	0
.400	-1	0	0	0
.420	-1	0	0	0
.440	-1	0	0	0
.460	-1	0	0	0
.480	-1	0	0	0
.500	-1	0	0	0
.520	-1	0	0	0
.540	-1	0	0	0
.560	-1	0	0	0
.580	-1	0	0	0
.600	1	0	0	0
.620	0	0	0	0
.640	0	0	0	0
.660	0	0	0	0
.680	0	0	0	0
.700	0	0	0	0
.720	0	0	0	0
.740	0	0	0	0
.760	0	0	0	0
.780	0	0	0	0
.800	0	0	0	0
.820	0	0	0	0
.840	0	0	0	0
.860	0	0	0	0
.880	0	0	0	0
.900	0	0	0	0
.920	0	0	0	0
.940	0	0	0	0
.960	0	0	0	0
.980	0	0	0	0
1.000	0	0	0	0

TABLE E-14 RECOVERY INDICATORS FOR COMPUTER A

TIME		RECOVERY B		
		RECOVERY REQUEST		
		A	B	C
.000	-1	0	0	0
.020	-1	0	0	0
.040	-1	0	0	0
.060	-1	0	0	0
.080	-1	0	0	0
.100	1	0	0	0
.120	0	0	0	0
.140	0	0	0	0
.160	0	0	0	0
.180	0	0	0	0
.200	4	0	0	1
.220	0	0	0	1
.240	0	0	0	0
.260	0	0	0	0
.280	0	0	0	0
.300	0	0	0	0
.320	0	0	0	0
.340	0	0	0	0
.360	0	0	0	0
.380	0	0	0	0
.400	0	0	0	0
.420	0	0	0	0
.440	0	0	0	0
.460	0	0	0	0
.480	0	0	0	0
.500	0	0	0	0
.520	0	0	0	0
.540	0	0	0	0
.560	0	0	0	0
.580	0	0	0	0
.600	3	0	0	0
.620	0	0	0	0
.640	0	0	0	0
.660	0	0	0	0
.680	0	0	0	0
.700	0	0	0	0
.720	0	0	0	0
.740	0	0	0	0
.760	0	0	0	0
.780	0	0	0	0
.800	0	0	0	0
.820	0	0	0	0
.840	0	0	0	0
.860	0	0	0	0
.880	0	0	0	0
.900	0	0	0	0
.920	0	0	0	0
.940	0	0	0	0
.960	0	0	0	0
.980	0	0	0	0
1.000	0	0	0	0

TABLE: E-15 RECOVERY INDICATORS FOR COMPUTER B.

ORIGINAL PAGE IS
OF POOR QUALITY

TIME		RECOVERY C		
		A	B	C
.000	-1	0	0	0
.020	-1	0	0	0
.040	-1	0	0	0
.060	-1	0	0	0
.080	-1	0	0	0
.100	-1	0	0	0
.120	-1	0	0	0
.140	-1	0	0	0
.160	-1	0	0	0
.180	-1	0	0	0
.200	1	0	0	1
.220	0	0	0	1
.240	0	0	0	0
.260	0	0	0	0
.280	0	0	0	0
.300	0	0	0	0
.320	0	0	0	0
.340	0	0	0	0
.360	0	0	0	0
.380	0	0	0	0
.400	0	0	0	0
.420	0	0	0	0
.440	0	0	0	0
.460	0	0	0	0
.480	0	0	0	0
.500	0	0	0	0
.520	0	0	0	0
.540	0	0	0	0
.560	0	0	0	0
.580	0	0	0	0
.600	4	0	0	0
.620	0	0	0	0
.640	0	0	0	0
.660	0	0	0	0
.680	0	0	0	0
.700	0	0	0	0
.720	0	0	0	0
.740	0	0	0	0
.760	0	0	0	0
.780	0	0	0	0
.800	0	0	0	0
.820	0	0	0	0
.840	0	0	0	0
.860	0	0	0	0
.880	0	0	0	0
.900	0	0	0	0
.920	0	0	0	0
.940	0	0	0	0
.960	0	0	0	0
.980	0	0	0	0
1.000	0	0	0	0

TABLE: E-16 RECOVERY INDICATORS FOR COMPUTER C.

ORIGINAL PAGE IS
OF POOR QUALITY

TIME	DO NOT USE			PERMANENT			T1 COUNT			T2 COUNT		
	A	B	C	A	B	C	A	B	C	A	B	C
.000	0	0	0	0	0	0	0	0	0	0	0	0
.020	0	0	0	0	1	1	0	0	0	0	0	0
.040	0	0	0	0	1	1	0	0	0	0	0	0
.060	0	0	0	0	1	1	0	0	0	0	0	0
.080	0	0	0	0	1	1	0	0	0	0	0	0
.100	0	1	0	0	0	1	0	1	0	0	1	0
.120	0	1	0	0	0	1	0	2	0	0	2	0
.140	0	1	0	0	0	1	0	3	0	0	3	0
.160	0	1	0	0	0	1	0	4	0	0	4	0
.180	0	1	0	0	0	1	0	5	0	0	5	0
.200	0	1	0	0	0	1	0	6	0	0	6	0
.220	0	0	0	0	0	1	0	0	0	0	0	0
.240	0	0	1	0	0	0	0	0	1	0	0	1
.260	0	0	1	0	0	0	0	0	2	0	0	2
.280	0	0	1	0	0	0	0	0	3	0	0	3
.300	0	0	1	0	0	0	0	0	4	0	0	4
.320	0	0	1	0	0	0	0	0	5	0	0	5
.340	0	0	1	0	0	0	0	0	6	0	0	6
.360	0	0	0	0	0	0	0	0	0	0	0	0
.380	0	0	0	0	0	0	0	0	0	0	0	0
.400	0	0	0	0	0	0	0	0	0	0	0	0
.420	0	0	0	0	0	0	0	0	0	0	0	0
.440	0	0	0	0	0	0	0	0	0	0	0	0
.460	0	0	0	0	0	0	0	0	0	0	0	0
.480	0	0	0	0	0	0	0	0	0	0	0	0
.500	0	0	0	0	0	0	0	0	0	0	0	0
.520	0	0	0	0	0	0	0	0	0	0	0	0
.540	0	0	0	0	0	0	0	0	0	0	0	0
.560	0	0	0	0	0	0	0	0	0	0	0	0
.580	0	0	0	0	0	0	0	0	0	0	0	0
.600	1	0	0	0	0	0	1	0	0	1	0	0
.620	1	0	0	0	0	0	2	0	0	2	0	0
.640	1	0	0	0	0	0	3	0	0	3	0	0
.660	1	0	0	0	0	0	4	0	0	4	0	0
.680	1	0	0	0	0	0	5	0	0	5	0	0
.700	1	0	0	0	0	0	6	0	0	6	0	0
.720	0	0	0	0	0	0	0	0	0	0	0	0
.740	0	0	0	0	0	0	0	0	0	0	0	0
.760	0	0	0	0	0	0	0	0	0	0	0	0
.780	0	0	0	0	0	0	0	0	0	0	0	0
.800	0	0	0	0	0	0	0	0	0	0	0	0
.820	0	0	0	0	0	0	0	0	0	0	0	0
.840	0	0	0	0	0	0	0	0	0	0	0	0
.860	0	0	0	0	0	0	0	0	0	0	0	0
.880	0	0	0	0	0	0	0	0	0	0	0	0
.900	0	0	0	0	0	0	0	0	0	0	0	0
.920	0	0	0	0	0	0	0	0	0	0	0	0
.940	0	0	0	0	0	0	0	0	0	0	0	0
.960	0	0	0	0	0	0	0	0	0	0	0	0
.980	0	0	0	0	0	0	0	0	0	0	0	0
1.000	0	0	0	0	0	0	0	0	0	0	0	0

TABLE E-17 DO NOT USE, PERMANENT FLAGS AND COUNTERS
FOR A, B, AND C SIGNALS FOR COMPUTER A

TIME	DO NOT USE			PERMANENT			T1 COUNT			T2 COUNT		
	A	B	C	A	B	C	A	B	C	A	B	C
.000	0	0	0	0	0	0	0	0	0	0	0	0
.020	0	0	0	0	0	0	0	0	0	0	0	0
.040	0	0	0	0	0	0	0	0	0	0	0	0
.060	0	0	0	0	0	0	0	0	0	0	0	0
.080	0	0	0	0	0	0	0	0	0	0	0	0
.100	0	1	0	0	0	1	0	1	0	0	1	0
.120	0	1	0	0	0	1	0	2	0	0	2	0
.140	0	1	0	0	0	1	0	3	0	0	3	0
.160	0	1	0	0	0	1	0	4	0	0	4	0
.180	0	1	0	0	0	1	0	5	0	0	5	0
.200	0	1	0	0	0	1	0	6	0	0	6	0
.220	0	1	0	0	0	1	0	0	0	0	0	0
.240	0	1	0	0	0	1	0	0	1	0	0	1
.260	0	0	1	0	0	0	0	0	2	0	0	2
.280	0	0	1	0	0	0	0	0	3	0	0	3
.300	0	0	1	0	0	0	0	0	4	0	0	4
.320	0	0	1	0	0	0	0	0	5	0	0	5
.340	0	0	1	0	0	0	0	0	6	0	0	6
.360	0	0	0	0	0	0	0	0	0	0	0	0
.380	0	0	0	0	0	0	0	0	0	0	0	0
.400	0	0	0	1	0	0	0	0	0	0	0	0
.420	0	0	0	1	0	0	0	0	0	0	0	0
.440	0	0	0	1	0	0	0	0	0	0	0	0
.460	0	0	1	0	0	0	0	0	0	0	0	0
.480	0	0	0	1	0	0	0	0	0	0	0	0
.500	0	0	0	1	0	0	0	0	0	0	0	0
.520	0	0	0	1	0	0	0	0	0	0	0	0
.540	0	0	0	1	0	0	0	0	0	0	0	0
.560	0	0	0	1	0	0	0	0	0	0	0	0
.580	0	0	0	1	0	0	0	0	0	0	0	0
.600	1	0	0	0	0	0	1	0	0	1	0	0
.620	1	0	0	0	0	0	2	0	0	2	0	0
.640	1	0	0	0	0	0	3	0	0	3	0	0
.660	1	0	0	0	0	0	4	0	0	4	0	0
.680	1	0	0	0	0	0	5	0	0	5	0	0
.700	1	0	0	0	0	0	6	0	0	6	0	0
.720	0	0	0	0	0	0	0	0	0	0	0	0
.740	0	0	0	0	0	0	0	0	0	0	0	0
.760	0	0	0	0	0	0	0	0	0	0	0	0
.780	0	0	0	0	0	0	0	0	0	0	0	0
.800	0	0	0	0	0	0	0	0	0	0	0	0
.820	0	0	0	0	0	0	0	0	0	0	0	0
.840	0	0	0	0	0	0	0	0	0	0	0	0
.860	0	0	0	0	0	0	0	0	0	0	0	0
.880	0	0	0	0	0	0	0	0	0	0	0	0
.900	0	0	0	0	0	0	0	0	0	0	0	0
.920	0	0	0	0	0	0	0	0	0	0	0	0
.940	0	0	0	0	0	0	0	0	0	0	0	0
.960	0	0	0	0	0	0	0	0	0	0	0	0
.980	0	0	0	0	0	0	0	0	0	0	0	0
1.000	0	0	0	0	0	0	0	0	0	0	0	0

TABLE E-18 DO NOT USE, PERMANENT FLAGS AND COUNTERS
FOR A, B, AND C SIGNALS FOR COMPUTER B

TIME	DO NOT USE			PERMANENT			T1 COUNT			T2 COUNT		
	A	B	C	A	B	C	A	B	C	A	B	C
.000	0	0	0	0	0	0	0	0	0	0	0	0
.020	0	0	0	0	0	0	0	0	0	0	0	0
.040	0	0	0	0	0	0	0	0	0	0	0	0
.060	0	0	0	0	0	0	0	0	0	0	0	0
.080	0	0	0	0	0	0	0	0	0	0	0	0
.100	0	0	0	0	0	0	0	0	0	0	0	0
.120	0	0	0	0	0	0	0	0	0	0	0	0
.140	0	0	0	0	0	0	0	0	0	0	0	0
.160	0	0	0	0	0	0	0	0	0	0	0	0
.180	0	0	0	0	0	0	0	0	0	0	0	0
.200	0	1	0	0	0	1	0	6	0	0	6	0
.220	0	0	0	0	0	1	0	0	0	0	0	0
.240	0	0	1	0	0	0	0	0	1	0	0	1
.260	0	0	1	0	0	0	0	0	2	0	0	2
.280	0	0	1	0	0	0	0	0	3	0	0	3
.300	0	0	1	0	0	0	0	0	4	0	0	4
.320	0	0	1	0	0	0	0	0	5	0	0	5
.340	0	0	1	0	0	0	0	0	6	0	0	6
.360	0	0	0	0	0	0	0	0	0	0	0	0
.380	0	0	0	0	0	0	0	0	0	0	0	0
.400	0	0	0	1	0	0	0	0	0	0	0	0
.420	0	0	0	1	0	0	0	0	0	0	0	0
.440	0	0	0	1	0	0	0	0	0	0	0	0
.460	0	0	0	1	0	0	0	0	0	0	0	0
.480	0	0	0	1	0	0	0	0	0	0	0	0
.500	0	0	0	1	0	0	0	0	0	0	0	0
.520	0	0	0	1	0	0	0	0	0	0	0	0
.540	0	0	0	1	0	0	0	0	0	0	0	0
.560	0	0	0	1	0	0	0	0	0	0	0	0
.580	0	0	0	1	0	0	0	0	0	0	0	0
.600	1	0	0	0	0	0	1	0	0	1	0	0
.620	1	0	0	0	0	0	2	0	0	2	0	0
.640	1	0	0	0	0	0	3	0	0	3	0	0
.660	1	0	0	0	0	0	4	0	0	4	0	0
.680	1	0	0	0	0	0	5	0	0	5	0	0
.700	1	0	0	0	0	0	6	0	0	6	0	0
.720	0	0	0	0	0	0	0	0	0	0	0	0
.740	0	0	0	0	0	0	0	0	0	0	0	0
.760	0	0	0	0	0	0	0	0	0	0	0	0
.780	0	0	0	0	0	0	0	0	0	0	0	0
.800	0	0	0	0	0	0	0	0	0	0	0	0
.820	0	0	0	0	0	0	0	0	0	0	0	0
.840	0	0	0	0	0	0	0	0	0	0	0	0
.860	0	0	0	0	0	0	0	0	0	0	0	0
.880	0	0	0	0	0	0	0	0	0	0	0	0
.900	0	0	0	0	0	0	0	0	0	0	0	0
.920	0	0	0	0	0	0	0	0	0	0	0	0
.940	0	0	0	0	0	0	0	0	0	0	0	0
.960	0	0	0	0	0	0	0	0	0	0	0	0
.980	0	0	0	0	0	0	0	0	0	0	0	0
1.000	0	0	0	0	0	0	0	0	0	0	0	0

TABLE: E-19 DO NOT USE, PERMANENT FLAGS AND COUNTERS
FOR A, B, AND C SIGNALS FOR COMPUTER C

TIME	SENSOR A	SELECTED SIGNAL	INPUT TO FILTER	OUTPUT OF FILTER
.000	.0000	.0000	.0000	.0000
.020	.1253	.1253	.1253	.0456
.040	.2487	.2487	.2487	.1 96
.060	.3681	.3681	.3681	.3068
.080	.4818	.4818	.4818	.4199
.100	.5878	.5878	.5878	.5285
.120	.6845	.6845	.6845	.6319
.140	.7705	.7705	.7705	.7241
.160	.8443	.8443	.8443	.8047
.180	.9048	.9048	.9048	.8728
.200	.9511	.9511	.9511	.9271
.220	.9823	.9823	.9823	.9668
.240	.9980	.9980	.9980	.9912
.260	.9980	.9980	.9980	1.0000
.280	.9823	.9823	.9823	.9931
.300	.9511	.9511	.9511	.9704
.320	.9048	.9048	.9048	.9325
.340	.8443	.8443	.8443	.8799
.360	.7705	.7705	.7705	.8133
.380	.6845	.6845	.6845	.7340
.400	.0000	.0000	.0000	.0000
.420	.0000	.0000	.0000	.0000
.440	.0000	.0000	.0000	.0000
.460	.0000	.0000	.0000	.0000
.480	.0000	.0000	.0000	.0000
.500	.0000	.0000	.0000	.0000
.520	.0000	.0000	.0000	.0000
.540	.0000	.0000	.0000	.0000
.560	.0000	.0000	.0000	.0000
.580	.0000	.0000	.0000	.0000
.600	-.5878	-.5878	-.5878	-.5297
.620	-.6845	-.6845	-.6845	-.6318
.640	-.7705	-.7705	-.7705	-.7240
.660	-.8443	-.8443	-.8443	-.8047
.680	-.9048	-.9048	-.9048	-.8728
.700	-.9511	-.9511	-.9511	-.9271
.720	-.9823	-.9823	-.9823	-.9668
.740	-.9980	-.9980	-.9980	-.9912
.760	-.9980	-.9980	-.9980	-1.0000
.780	-.9823	-.9823	-.9823	-.9931
.800	-.9511	-.9511	-.9511	-.9704
.820	-.9048	-.9048	-.9048	-.9325
.840	-.8443	-.8443	-.8443	-.8799
.860	-.7705	-.7705	-.7705	-.8133
.880	-.6845	-.6845	-.6845	-.7340
.900	-.5878	-.5878	-.5878	-.6431
.920	-.4818	-.4818	-.4818	-.5420
.940	-.3681	-.3681	-.3681	-.4324
.960	-.2487	-.2487	-.2487	-.3160
.980	-.1253	-.1253	-.1253	-.1946
1.000	.0000	.0000	.0000	-.0701

TABLE:E-20 INPUT AND OUTPUT FOR COMPUTER A

TIME	SENSOR B	SELECTED SIGNAL	INPUT TO FILTERS	OUTPUT OF FILTER
.000	.0000	.0000	.0000	.0000
.020	.0000	.0000	.0000	.0000
.040	.0000	.0000	.0000	.0000
.060	.0000	.0000	.0000	.0000
.080	.0000	.0000	.0000	.0000
.100	.5878	.5878	.5878	.5286
.120	.6845	.6845	.6845	.6319
.140	.7705	.7705	.7705	.7241
.160	.8443	.8443	.8443	.8047
.180	.9048	.9048	.9048	.8728
.200	.9511	.9511	.9511	.9271
.220	.9823	.9823	.9823	.9668
.240	.9980	.9980	.9980	.9912
.260	.9980	.9980	.9980	1.0000
.280	.9823	.9823	.9823	.9931
.300	.9511	.9511	.9511	.9704
.320	.9048	.9048	.9048	.9325
.340	.8443	.8443	.8443	.8799
.360	.7705	.7705	.7705	.8133
.380	.6845	.6845	.6845	.7340
.400	.5878	.5878	.5878	.6431
.420	.4818	.4818	.4818	.5420
.440	.3681	.3681	.3681	.4324
.460	.2487	.2487	.2487	.3160
.480	.1253	.1253	.1253	.1946
.500	-.0000	-.0000	-.0000	.0701
.520	-.1253	-.1253	-.1253	-.0555
.540	-.2487	-.2487	-.2487	-.1802
.560	-.3681	-.3681	-.3681	-.3021
.580	-.4818	-.4818	-.4818	-.4192
.600	-.5878	-.5878	-.5878	-.5297
.620	-.6845	-.6845	-.6845	-.6318
.640	-.7705	-.7705	-.7705	-.7240
.660	-.8443	-.8443	-.8443	-.8047
.680	-.9048	-.9048	-.9048	-.8728
.700	-.9511	-.9511	-.9511	-.9271
.720	-.9823	-.9823	-.9823	-.9668
.740	-.9980	-.9980	-.9980	-.9912
.760	-.9980	-.9980	-.9980	-1.0000
.780	-.9823	-.9823	-.9823	-.9931
.800	-.9511	-.9511	-.9511	-.9704
.820	-.9048	-.9048	-.9048	-.9325
.840	-.8443	-.8443	-.8443	-.8799
.860	-.7705	-.7705	-.7705	-.8133
.880	-.6845	-.6845	-.6845	-.7340
.900	-.5878	-.5878	-.5878	-.6431
.920	-.4818	-.4818	-.4818	-.5420
.940	-.3681	-.3681	-.3681	-.4324
.960	-.2487	-.2487	-.2487	-.3160
.980	-.1253	-.1253	-.1253	-.1946
1.000	.0000	.0000	.0000	-.0701

TABLE E-21 INPUT AND OUTPUT FOR COMPUTER B

TIME	SENSOR C	SELECTED SIGNAL	INPUT TO FILTER	OUTPUT OF FILTER
.000	.0000	.0000	.0000	.0000
.020	.0000	.0000	.0000	.0000
.040	.0000	.0000	.0000	.0000
.060	.0000	.0000	.0000	.0000
.080	.0000	.0000	.0000	.0000
.100	.0000	.0000	.0000	.0000
.120	.0000	.0000	.0000	.0000
.140	.0000	.0000	.0000	.0000
.160	.0000	.0000	.0000	.0000
.180	.0000	.0000	.0000	.0000
.200	.9511	.9511	.9511	.9271
.220	.9823	.9823	.9823	.9668
.240	.9980	.9980	.9980	.9912
.260	.9980	.9980	.9980	1.0000
.280	.9823	.9823	.9823	.9931
.300	.9511	.9511	.9511	.9704
.320	.9048	.9048	.9048	.9325
.340	.8443	.8443	.8443	.8799
.360	.7705	.7705	.7705	.8133
.380	.6845	.6845	.6845	.7340
.400	.5878	.5878	.5878	.6431
.420	.4818	.4818	.4818	.5420
.440	.3681	.3681	.3681	.4324
.460	.2487	.2487	.2487	.3160
.480	.1253	.1253	.1253	.1946
.500	-.0000	-.0000	-.0000	.0701
.520	-.1253	-.1253	-.1253	-.0555
.540	-.2487	-.2487	-.2487	-.1802
.560	-.3681	-.3681	-.3681	-.3021
.580	-.4818	-.4818	-.4818	-.4192
.600	-.5878	-.5878	-.5878	-.5297
.620	-.6845	-.6845	-.6845	-.6318
.640	-.7705	-.7705	-.7705	-.7240
.660	-.8443	-.8443	-.8443	-.8047
.680	-.9048	-.9048	-.9048	-.8728
.700	-.9511	-.9511	-.9511	-.9271
.720	-.9823	-.9823	-.9823	-.9668
.740	-.9980	-.9980	-.9980	-.9912
.760	-.9980	-.9980	-.9980	-1.0000
.780	-.9823	-.9823	-.9823	-.9931
.800	-.9511	-.9511	-.9511	-.9704
.820	-.9048	-.9048	-.9048	-.9325
.840	-.8443	-.8443	-.8443	-.8799
.860	-.7705	-.7705	-.7705	-.8133
.880	-.6845	-.6845	-.6845	-.7340
.900	-.5878	-.5878	-.5878	-.6431
.920	-.4818	-.4818	-.4818	-.5420
.940	-.3681	-.3681	-.3681	-.4324
.960	-.2487	-.2487	-.2487	-.3160
.980	-.1253	-.1253	-.1253	-.1946
1.000	.0000	.0000	.0000	-.0701

TABLE E-22 INPUT AND OUTPUT FOR COMPUTER C

Double Power Fault

After attaining triplex operation the A computer experiences a power loss from 0.4 to 0.48 seconds. The C computer then has power loss from 0.6 to 0.68 seconds. This can be seen by examining Table E-23.

The synchronization status for the system is shown in Table E-24. Tables E-25, E-26, and E-27 show the recovery requests as seen by each computer. At 0.02 computer A comes on notes it is alone and proceeds in simplex. At 0.1 A sees B come up and releases B's permanent flags and sets its do-not-use flags. At 0.2 A see C come up, but since B has not completed recovery yet holds C failed until 0.22 when B finally recovers. From 0.4 to 0.48 A's power is off. At 0.5 A's power comes back on and A recovers to C. At 0.7 A releases C for recovery again.

At 0.1 B comes on notes that A is operating and recovers to A. Then after recovering B remains good during the entire run.

At 0.2 C comes on and recovers to B, since B's do-not-use flags are still set, C does not begin recover until B has completed recovery.

Tables E-28, E-29, and E-30 show the do-not-use flags permanent flags, T1 counter, and T2 counter for A, B, and C respectively.

The inputs and outputs for A, B, and C are shown in Tables E-31, E-32, and E-33.

TIME	POWER			WATCH-DOG			POWER INTERUPT			ITERATION RESET		
	A	B	C	A	B	C	A	B	C	A	B	C
.0000	F	F	F	F	F	F	F	F	F	F	F	F
.0200	T	F	F	T	F	F	T	F	F	T	F	F
.0400	T	F	F	T	F	F	F	F	F	T	F	F
.0600	T	F	F	T	F	F	F	F	F	T	F	F
.0800	T	F	F	T	F	F	F	F	F	T	F	F
.1000	T	T	F	T	T	F	F	T	F	T	T	F
.1200	T	T	F	T	T	F	F	F	F	T	T	F
.1400	T	T	F	T	T	F	F	F	F	T	T	F
.1600	T	T	F	T	T	F	F	F	F	T	T	F
.1800	T	T	F	T	T	F	F	F	F	T	T	F
.2000	T	T	T	T	T	T	F	F	T	T	T	T
.2200	T	T	T	T	T	T	F	F	F	T	T	T
.2400	T	T	T	T	T	T	F	F	F	T	T	T
.2600	T	T	T	T	T	T	F	F	F	T	T	T
.2800	T	T	T	T	T	T	F	F	F	T	T	T
.3000	T	T	T	T	T	T	F	F	F	T	T	T
.3200	T	T	T	T	T	T	F	F	F	T	T	T
.3400	T	T	T	T	T	T	F	F	F	T	T	T
.3600	T	T	T	T	T	T	F	F	F	T	T	T
.3800	T	T	T	T	T	T	F	F	F	T	T	T
.4000	F	T	T	F	T	T	F	F	F	T	T	T
.4200	F	T	T	F	T	T	F	F	F	F	T	T
.4400	F	T	T	F	T	T	F	F	F	F	T	T
.4600	F	T	T	F	T	T	F	F	F	F	T	T
.4800	F	T	T	F	T	T	F	F	F	F	T	T
.5000	T	T	T	T	T	T	T	F	F	T	T	T
.5200	T	T	T	T	T	T	F	F	F	T	T	T
.5400	T	T	T	T	T	T	F	F	F	T	T	T
.5600	T	T	T	T	T	T	F	F	F	T	T	T
.5800	T	T	T	T	T	T	F	F	F	T	T	T
.6000	T	T	F	T	T	F	F	F	F	T	T	T
.6200	T	T	F	T	T	F	F	F	F	T	T	F
.6400	T	T	F	T	T	F	F	F	F	T	T	F
.6600	T	T	F	T	T	F	F	F	F	T	T	F
.6800	T	T	F	T	T	F	F	F	F	T	T	F
.7000	T	T	T	T	T	T	F	F	T	T	T	T
.7200	T	T	T	T	T	T	F	F	F	T	T	T
.7400	T	T	T	T	T	T	F	F	F	T	T	T
.7600	T	T	T	T	T	T	F	F	F	T	T	T
.7800	T	T	T	T	T	T	F	F	F	T	T	T
.8000	T	T	T	T	T	T	F	F	F	T	T	T
.8200	T	T	T	T	T	T	F	F	F	T	T	T
.8400	T	T	T	T	T	T	F	F	F	T	T	T
.8600	T	T	T	T	T	T	F	F	F	T	T	T
.8800	T	T	T	T	T	T	F	F	F	T	T	T
.9000	T	T	T	T	T	T	F	F	F	T	T	T
.9200	T	T	T	T	T	T	F	F	F	T	T	T
.9400	T	T	T	T	T	T	F	F	F	T	T	T
.9600	T	T	T	T	T	T	F	F	F	T	T	T
.9800	T	T	T	T	T	T	F	F	F	T	T	T
1.0000	T	T	T	T	T	T	F	F	F	T	T	T

TABLE E-23 SIMULATED HARDWARE INDICATORS

TIME	A	B	C
.000	0	0	0
.020	4	0	0
.040	4	0	0
.060	4	0	0
.080	4	0	0
.100	6	6	0
.120	6	6	0
.140	6	6	0
.160	6	6	0
.180	6	6	0
.200	7	7	7
.220	7	7	7
.240	7	7	7
.260	7	7	7
.280	7	7	7
.300	7	7	7
.320	7	7	7
.340	7	7	7
.360	7	7	7
.380	7	7	7
.400	0	3	3
.420	0	3	3
.440	0	3	3
.460	0	3	3
.480	0	3	3
.500	7	7	7
.520	7	7	7
.540	7	7	7
.560	7	7	7
.580	7	7	7
.600	6	6	0
.620	6	6	0
.640	6	6	0
.660	6	6	0
.680	6	6	0
.700	7	7	7
.720	7	7	7
.740	7	7	7
.760	7	7	7
.780	7	7	7
.800	7	7	7
.820	7	7	7
.840	7	7	7
.860	7	7	7
.880	7	7	7
.900	7	7	7
.920	7	7	7
.940	7	7	7
.960	7	7	7
.980	7	7	7
1.000	7	7	7

TABLE: E-24 SYNCHRONIZATION INDICATORS

TIME	RECOVERY A		RECOVERY REQUEST	
		A	B	C
.000	-1	0	0	0
.020	1	0	0	0
.040	0	0	0	0
.060	0	0	0	0
.080	0	0	0	0
.100	0	0	0	0
.120	0	0	0	0
.140	0	0	0	0
.160	0	0	0	0
.180	0	0	0	0
.200	3	0	0	1
.220	0	0	0	1
.240	0	0	0	0
.260	0	0	0	0
.280	0	0	0	0
.300	0	0	0	0
.320	0	0	0	0
.340	0	0	0	0
.360	0	0	0	0
.380	0	0	0	0
.400	-1	0	0	0
.420	-1	0	0	0
.440	-1	0	0	0
.460	-1	0	0	0
.480	-1	0	0	0
.500	1	0	0	0
.520	0	0	0	0
.540	0	0	0	0
.560	0	0	0	0
.580	0	0	0	0
.600	0	0	0	0
.620	0	0	0	0
.640	0	0	0	0
.660	0	0	0	0
.680	0	0	0	0
.700	3	0	0	0
.720	0	0	0	0
.740	0	0	0	0
.760	0	0	0	0
.780	0	0	0	0
.800	0	0	0	0
.820	0	0	0	0
.840	0	0	0	0
.860	0	0	0	0
.880	0	0	0	0
.900	0	0	0	0
.920	0	0	0	0
.940	0	0	0	0
.960	0	0	0	0
.980	0	0	0	0
1.000	0	0	0	0

TABLE E-25 RECOVERY INDICATORS FOR COMPUTER A

TIME		RECOVERY B		RECOVERY REQUEST	
		A	B	C	
.000	-1	0	0	0	
.020	-1	0	0	0	
.040	-1	0	0	0	
.060	-1	0	0	0	
.080	-1	0	0	0	
.100	1	0	0	0	
.120	0	0	0	0	
.140	0	0	0	0	
.160	0	0	0	0	
.180	0	0	0	0	
.200	4	0	0	1	
.220	0	0	0	1	
.240	0	0	0	0	
.260	0	0	0	0	
.280	0	0	0	0	
.300	0	0	0	0	
.320	0	0	0	0	
.340	0	0	0	0	
.360	0	0	0	0	
.380	0	0	0	0	
.400	0	0	0	0	
.420	0	0	0	0	
.440	0	0	0	0	
.460	0	0	0	0	
.480	0	0	0	0	
.500	3	0	0	0	
.520	0	0	0	0	
.540	0	0	0	0	
.560	0	0	0	0	
.580	0	0	0	0	
.600	0	0	0	0	
.620	0	0	0	0	
.640	0	0	0	0	
.660	0	0	0	0	
.680	0	0	0	0	
.700	4	0	0	0	
.720	0	0	0	0	
.740	0	0	0	0	
.760	0	0	0	0	
.780	0	0	0	0	
.800	0	0	0	0	
.820	0	0	0	0	
.840	0	0	0	0	
.860	0	0	0	0	
.880	0	0	0	0	
.900	0	0	0	0	
.920	0	0	0	0	
.940	0	0	0	0	
.960	0	0	0	0	
.980	0	0	0	0	
1.000	0	0	0	0	

TIME		RECOVERY C		RECOVERY REQUEST	
		A	B	C	
.000	-1	0	0	0	
.020	-1	0	0	0	
.040	-1	0	0	0	
.060	-1	0	0	0	
.080	-1	0	0	0	
.100	-1	0	0	0	
.120	-1	0	0	0	
.140	-1	0	0	0	
.160	-1	0	0	0	
.180	-1	0	0	0	
.200	1	0	0	1	
.220	0	0	0	1	
.240	0	0	0	0	
.260	0	0	0	0	
.280	0	0	0	0	
.300	0	0	0	0	
.320	0	0	0	0	
.340	0	0	0	0	
.360	0	0	0	0	
.380	0	0	0	0	
.400	0	0	0	0	
.420	0	0	0	0	
.440	0	0	0	0	
.460	0	0	0	0	
.480	0	0	0	0	
.500	4	0	0	0	
.520	0	0	0	0	
.540	0	0	0	0	
.560	0	0	0	0	
.580	0	0	0	0	
.600	-1	0	0	0	
.620	-1	0	0	0	
.640	-1	0	0	0	
.660	-1	0	0	0	
.680	-1	0	0	0	
.700	1	0	0	0	
.720	0	0	0	0	
.740	0	0	0	0	
.760	0	0	0	0	
.780	0	0	0	0	
.800	0	0	0	0	
.820	0	0	0	0	
.840	0	0	0	0	
.860	0	0	0	0	
.880	0	0	0	0	
.900	0	0	0	0	
.920	0	0	0	0	
.940	0	0	0	0	
.960	0	0	0	0	
.980	0	0	0	0	
1.000	0	0	0	0	

TABLE: E-27 RECOVERY INDICATORS FOR COMPUTER C.

TIME	DO NOT USE			PERMANENT			T1 COUNT			T2 COUNT		
	A	B	C	A	B	C	A	B	C	A	B	C
.000	0	0	0	0	0	0	0	0	0	0	0	0
.020	0	0	0	0	1	1	0	0	0	0	0	0
.040	0	0	0	0	1	1	0	0	0	0	0	0
.060	0	0	0	0	1	1	0	0	0	0	0	0
.080	0	0	0	0	1	1	0	0	0	0	0	0
.100	0	1	0	0	0	1	0	1	0	0	1	0
.120	0	1	0	0	0	1	0	2	0	0	2	0
.140	0	1	0	0	0	1	0	3	0	0	3	0
.160	0	1	0	0	0	1	0	4	0	0	4	0
.180	0	1	0	0	0	1	0	5	0	0	5	0
.200	0	1	0	0	0	1	0	6	0	0	6	0
.220	0	0	0	0	0	1	0	0	0	0	0	0
.240	0	0	1	0	0	0	0	0	1	0	0	1
.260	0	0	1	0	0	0	0	0	2	0	0	2
.280	0	0	1	0	0	0	0	0	3	0	0	3
.300	0	0	1	0	0	0	0	0	4	0	0	4
.320	0	0	1	0	0	0	0	0	5	0	0	5
.340	0	0	1	0	0	0	0	0	6	0	0	6
.360	0	0	0	0	0	0	0	0	0	0	0	0
.380	0	0	0	0	0	0	0	0	0	0	0	0
.400												
.420												
.440												
.460												
.480												
.500	1	0	0	0	0	0	1	0	0	1	0	0
.520	1	0	0	0	0	0	2	0	0	2	0	0
.540	1	0	0	0	0	0	3	0	0	3	0	0
.560	1	0	0	0	0	0	4	0	0	4	0	0
.580	1	0	0	0	0	0	5	0	0	5	0	0
.600	1	0	0	0	0	1	6	0	0	6	0	0
.620	0	0	0	0	0	1	0	0	0	0	0	0
.640	0	0	0	0	0	1	0	0	0	0	0	0
.660	0	0	0	0	0	1	0	0	0	0	0	0
.680	0	0	0	0	0	1	0	0	0	0	0	0
.700	0	0	1	0	0	0	0	0	1	0	0	1
.720	0	0	1	0	0	0	0	0	2	0	0	2
.740	0	0	1	0	0	0	0	0	3	0	0	3
.760	0	0	1	0	0	0	0	0	4	0	0	4
.780	0	0	1	0	0	0	0	0	5	0	0	5
.800	0	0	1	0	0	0	0	0	6	0	0	6
.820	0	0	0	0	0	0	0	0	0	0	0	0
.840	0	0	0	0	0	0	0	0	0	0	0	0
.860	0	0	0	0	0	0	0	0	0	0	0	0
.880	0	0	0	0	0	0	0	0	0	0	0	0
.900	0	0	0	0	0	0	0	0	0	0	0	0
.920	0	0	0	0	0	0	0	0	0	0	0	0
.940	0	0	0	0	0	0	0	0	0	0	0	0
.960	0	0	0	0	0	0	0	0	0	0	0	0
.980	0	0	0	0	0	0	0	0	0	0	0	0
1.000	0	0	0	0	0	0	0	0	0	0	0	0

A is off

TABLE: E-28: DO NOT USE, PERMANENT FLAGS AND COUNTERS.
FOR A, B, AND C SIGNALS FOR COMPUTER A

TIME	DO NOT USE			PERMANENT			T1 COUNT			T2 COUNT		
	A	B	C	A	B	C	A	B	C	A	B	C
.000	0	0	0	0	0	0	0	0	0	0	0	0
.020	0	0	0	0	0	0	0	0	0	0	0	0
.040	0	0	0	0	0	0	0	0	0	0	0	0
.060	0	0	0	0	0	0	0	0	0	0	0	0
.080	0	0	0	0	0	0	0	0	0	0	0	0
.100	0	1	0	0	0	1	0	1	0	0	1	0
.120	0	1	0	0	0	1	0	2	0	0	2	0
.140	0	1	0	0	0	1	0	3	0	0	3	0
.160	0	1	0	0	0	1	0	4	0	0	4	0
.180	0	1	0	0	0	1	0	5	0	0	5	0
.200	0	1	0	0	0	1	0	6	0	0	6	0
.220	0	0	0	0	0	1	0	0	0	0	0	0
.240	0	0	1	0	0	0	0	0	1	0	0	1
.260	0	0	1	0	0	0	0	0	2	0	0	2
.280	0	0	1	0	0	0	0	0	3	0	0	3
.300	0	0	1	0	0	0	0	0	4	0	0	4
.320	0	0	1	0	0	0	0	0	5	0	0	5
.340	0	0	1	0	0	0	0	0	6	0	0	6
.360	0	0	0	0	0	0	0	0	0	0	0	0
.380	0	0	0	0	0	0	0	0	0	0	0	0
.400	0	0	0	1	0	0	0	0	0	0	0	0
.420	0	0	0	1	0	0	0	0	0	0	0	0
.440	0	0	0	1	0	0	0	0	0	0	0	0
.460	0	0	0	1	0	0	0	0	0	0	0	0
.480	0	0	0	1	0	0	0	0	0	0	0	0
.500	1	0	0	0	0	0	1	0	0	1	0	0
.520	1	0	0	0	0	0	2	0	0	2	0	0
.540	1	0	0	0	0	0	3	0	0	3	0	0
.560	1	0	0	0	0	0	4	0	0	4	0	0
.580	1	0	0	0	0	0	5	0	0	5	0	0
.600	1	0	0	0	0	1	6	0	0	6	0	0
.620	0	0	0	0	0	1	0	0	0	0	0	0
.640	0	0	0	0	0	1	0	0	0	0	0	0
.660	0	0	0	0	0	1	0	0	0	0	0	0
.680	0	0	0	0	0	1	0	0	0	0	0	0
.700	0	0	1	0	0	0	0	0	1	0	0	1
.720	0	0	1	0	0	0	0	0	2	0	0	2
.740	0	0	1	0	0	0	0	0	3	0	0	3
.760	0	0	1	0	0	0	0	0	4	0	0	4
.780	0	0	1	0	0	0	0	0	5	0	0	5
.800	0	0	1	0	0	0	0	0	6	0	0	6
.820	0	0	0	0	0	0	0	0	0	0	0	0
.840	0	0	0	0	0	0	0	0	0	0	0	0
.860	0	0	0	0	0	0	0	0	0	0	0	0
.880	0	0	0	0	0	0	0	0	0	0	0	0
.900	0	0	0	0	0	0	0	0	0	0	0	0
.920	0	0	0	0	0	0	0	0	0	0	0	0
.940	0	0	0	0	0	0	0	0	0	0	0	0
.960	0	0	0	0	0	0	0	0	0	0	0	0
.980	0	0	0	0	0	0	0	0	0	0	0	0
1.000	0	0	0	0	0	0	0	0	0	0	0	0

TABLE: E-29 DO NOT USE, PERMANENT FLAGS AND COUNTERS.
FOR A, B, AND C SIGNALS FOR COMPUTER B.

TIME	DO NOT USE			PERMANENT			T1 COUNT			T2 COUNT		
	A	B	C	A	B	C	A	B	C	A	B	C
.000	0	0	0	0	0	0	0	0	0	0	0	0
.020	0	0	0	0	0	0	0	0	0	0	0	0
.040	0	0	0	0	0	0	0	0	0	0	0	0
.060	0	0	0	0	0	0	0	0	0	0	0	0
.080	0	0	0	0	0	0	0	0	0	0	0	0
.100	0	0	0	0	0	0	0	0	0	0	0	0
.120	0	0	0	0	0	0	0	0	0	0	0	0
.140	0	0	0	0	0	0	0	0	0	0	0	0
.160	0	0	0	0	0	0	0	0	0	0	0	0
.180	0	0	0	0	0	0	0	0	0	0	0	0
.200	0	1	0	0	0	1	0	6	0	0	6	0
.220	0	0	0	0	0	1	0	0	0	0	0	0
.240	0	0	1	0	0	0	0	0	1	0	0	1
.260	0	0	1	0	0	0	0	0	2	0	0	2
.280	0	0	1	0	0	0	0	0	3	0	0	3
.300	0	0	1	0	0	0	0	0	4	0	0	4
.320	0	0	1	0	0	0	0	0	5	0	0	5
.340	0	0	1	0	0	0	0	0	6	0	0	6
.360	0	0	0	0	0	0	0	0	0	0	0	0
.380	0	0	0	0	0	0	0	0	0	0	0	0
.400	0	0	0	1	0	0	0	0	0	0	0	0
.420	0	0	0	1	0	0	0	0	0	0	0	0
.440	0	0	0	1	0	0	0	0	0	0	0	0
.460	0	0	0	1	0	0	0	0	0	0	0	0
.480	0	0	0	1	0	0	0	0	0	0	0	0
.500	1	0	0	0	0	0	1	0	0	1	0	0
.520	1	0	0	0	0	0	2	0	0	2	0	0
.540	1	0	0	0	0	0	3	0	0	3	0	0
.560	1	0	0	0	0	0	4	0	0	4	0	0
.580	1	0	0	0	0	0	5	0	0	5	0	0
.600												
.620												
.640												
.660												
.680												
.700	0	0	1	0	0	0	0	0	1	0	0	1
.720	0	0	1	0	0	0	0	0	2	0	0	2
.740	0	0	1	0	0	0	0	0	3	0	0	3
.760	0	0	1	0	0	0	0	0	4	0	0	4
.780	0	0	1	0	0	0	0	0	5	0	0	5
.800	0	0	1	0	0	0	0	0	6	0	0	6
.820	0	0	0	0	0	0	0	0	0	0	0	0
.840	0	0	0	0	0	0	0	0	0	0	0	0
.860	0	0	0	0	0	0	0	0	0	0	0	0
.880	0	0	0	0	0	0	0	0	0	0	0	0
.900	0	0	0	0	0	0	0	0	0	0	0	0
.920	0	0	0	0	0	0	0	0	0	0	0	0
.940	0	0	0	0	0	0	0	0	0	0	0	0
.960	0	0	0	0	0	0	0	0	0	0	0	0
.980	0	0	0	0	0	0	0	0	0	0	0	0
1.000	0	0	0	0	0	0	0	0	0	0	0	0

C is off

TABLE: E-30 DO NOT USE, PERMANENT FLAGS AND COUNTERS
FOR A, B, AND C SIGNALS FOR COMPUTER C

TIME	SENSOR A	SELECTED SIGNAL	INPUT TO FILTER	OUTPUT OF FILTERS
.000	.0000	.0000	.0000	.0000
.020	.1253	.1253	.1253	.0456
.040	.2487	.2487	.2487	.1696
.060	.3681	.3681	.3681	.3068
.080	.4818	.4818	.4818	.4199
.100	.5878	.5878	.5878	.5286
.120	.6845	.6845	.6845	.6319
.140	.7705	.7705	.7705	.7241
.160	.8443	.8443	.8443	.8047
.180	.9048	.9048	.9048	.8728
.200	.9511	.9511	.9511	.9271
.220	.9823	.9823	.9823	.9668
.240	.9980	.9980	.9980	.9912
.260	.9980	.9980	.9980	1.0000
.280	.9823	.9823	.9823	.9931
.300	.9511	.9511	.9511	.9704
.320	.9048	.9048	.9048	.9325
.340	.8443	.8443	.8443	.8799
.360	.7705	.7705	.7705	.8133
.380	.6845	.6845	.6845	.7340
.400				
.420				
.440				
.460				
.480				
.500	-.0000	-.0000	-.0000	.0701
.520	-.1253	-.1253	-.1253	-.0555
.540	-.2487	-.2487	-.2487	-.1802
.560	-.3681	-.3681	-.3681	-.3021
.580	-.4818	-.4818	-.4818	-.4192
.600	-.5878	-.5878	-.5878	-.5297
.620	-.6845	-.6845	-.6845	-.6318
.640	-.7705	-.7705	-.7705	-.7240
.660	-.8443	-.8443	-.8443	-.8047
.680	-.9048	-.9048	-.9048	-.8728
.700	-.9511	-.9511	-.9511	-.9271
.720	-.9823	-.9823	-.9823	-.9668
.740	-.9980	-.9980	-.9980	-.9912
.760	-.9980	-.9980	-.9980	-1.0000
.780	-.9823	-.9823	-.9823	-.9931
.800	-.9511	-.9511	-.9511	-.9704
.820	-.9048	-.9048	-.9048	-.9325
.840	-.8443	-.8443	-.8443	-.8799
.860	-.7705	-.7705	-.7705	-.8133
.880	-.6845	-.6845	-.6845	-.7340
.900	-.5878	-.5878	-.5878	-.6431
.920	-.4818	-.4818	-.4818	-.5420
.940	-.3681	-.3681	-.3681	-.4324
.960	-.2487	-.2487	-.2487	-.3160
.980	-.1253	-.1253	-.1253	-.1946
1.000	.0000	.0000	.0000	-.0701

} A is off

TABLE: E-31 INPUT AND OUTPUT FOR COMPUTER A

TIME	SENSOR B	SELECTED SIGNAL	INPUT TO FILTER	OUTPUT OF FILTER
.000	.0000	.0000	.0000	.0000
.020	.0000	.0000	.0000	.0000
.040	.0000	.0000	.0000	.0000
.060	.0000	.0000	.0000	.0000
.080	.0000	.0000	.0000	.0000
.100	.5878	.5878	.5878	.5286
.120	.6845	.6845	.6845	.6319
.140	.7705	.7705	.7705	.7241
.160	.8443	.8443	.8443	.8047
.180	.9048	.9048	.9048	.8728
.200	.9511	.9511	.9511	.9271
.220	.9823	.9823	.9823	.9668
.240	.9980	.9980	.9980	.9912
.260	.9980	.9980	.9980	1.0000
.280	.9823	.9823	.9823	.9931
.300	.9511	.9511	.9511	.9704
.320	.9048	.9048	.9048	.9325
.340	.8443	.8443	.8443	.8799
.360	.7705	.7705	.7705	.8133
.380	.6845	.6845	.6845	.7340
.400	.5878	.5878	.5878	.6431
.420	.4818	.4818	.4818	.5420
.440	.3681	.3681	.3681	.4324
.460	.2487	.2487	.2487	.3160
.480	.1253	.1253	.1253	.1946
.500	-.0000	-.0000	-.0000	.0701
.520	-.1253	-.1253	-.1253	-.0555
.540	-.2487	-.2487	-.2487	-.1802
.560	-.3681	-.3681	-.3681	-.3021
.580	-.4818	-.4818	-.4818	-.4192
.600	-.5878	-.5878	-.5878	-.5297
.620	-.6845	-.6845	-.6845	-.6318
.640	-.7705	-.7705	-.7705	-.7240
.660	-.8443	-.8443	-.8443	-.8047
.680	-.9048	-.9048	-.9048	-.8728
.700	-.9511	-.9511	-.9511	-.9271
.720	-.9823	-.9823	-.9823	-.9668
.740	-.9980	-.9980	-.9980	-.9912
.760	-.9980	-.9980	-.9980	-1.0000
.780	-.9823	-.9823	-.9823	-.9931
.800	-.9511	-.9511	-.9511	-.9704
.820	-.9048	-.9048	-.9048	-.9325
.840	-.8443	-.8443	-.8443	-.8799
.860	-.7705	-.7705	-.7705	-.8133
.880	-.6845	-.6845	-.6845	-.7340
.900	-.5878	-.5878	-.5878	-.6431
.920	-.4818	-.4818	-.4818	-.5420
.940	-.3681	-.3681	-.3681	-.4324
.960	-.2487	-.2487	-.2487	-.3160
.980	-.1253	-.1253	-.1253	-.1946
1.000	.0000	.0000	.0000	-.0701

TABLE: E-32 INPUT AND OUTPUT FOR COMPUTER B

TIME	SENSOR C	SELECTED SIGNAL	INPUT TO FILTER	OUTPUT OF FILTER
.000	.0000	.0000	.0000	.0000
.020	.0000	.0000	.0000	.0000
.040	.0000	.0000	.0000	.0000
.060	.0000	.0000	.0000	.0000
.080	.0000	.0000	.0000	.0000
.100	.0000	.0000	.0000	.0000
.120	.0000	.0000	.0000	.0000
.140	.0000	.0000	.0000	.0000
.160	.0000	.0000	.0000	.0000
.180	.0000	.0000	.0000	.0000
.200	.9511	.9511	.9511	.9271
.220	.9823	.9823	.9823	.9668
.240	.9980	.9980	.9980	.9912
.260	.9980	.9980	.9980	1.0000
.280	.9823	.9823	.9823	.9931
.300	.9511	.9511	.9511	.9704
.320	.9048	.9048	.9048	.9325
.340	.8443	.8443	.8443	.8799
.360	.7705	.7705	.7705	.8133
.380	.6845	.6845	.6845	.7340
.400	.5878	.5878	.5878	.6431
.420	.4818	.4818	.4818	.5420
.440	.3681	.3681	.3681	.4324
.460	.2487	.2487	.2487	.3160
.480	.1253	.1253	.1253	.1946
.500	-.0000	-.0000	-.0000	-.0701
.520	-.1253	-.1253	-.1253	-.0555
.540	-.2487	-.2487	-.2487	-.1802
.560	-.3681	-.3681	-.3681	-.3021
.580	-.4818	-.4818	-.4818	-.4192
.600				
.620				
.640				
.660				
.680				
.700	-.9511	-.9511	-.9511	-.9271
.720	-.9823	-.9823	-.9823	-.9668
.740	-.9980	-.9980	-.9980	-.9912
.760	-.9980	-.9980	-.9980	-1.0000
.780	-.9823	-.9823	-.9823	-.9931
.800	-.9511	-.9511	-.9511	-.9704
.820	-.9048	-.9048	-.9048	-.9325
.840	-.8443	-.8443	-.8443	-.8799
.860	-.7705	-.7705	-.7705	-.8133
.880	-.6845	-.6845	-.6845	-.7340
.900	-.5878	-.5878	-.5878	-.6431
.920	-.4818	-.4818	-.4818	-.5420
.940	-.3681	-.3681	-.3681	-.4324
.960	-.2487	-.2487	-.2487	-.3160
.980	-.1253	-.1253	-.1253	-.1946
1.000	.0000	.0000	.0000	-.0701

C is off

TABLE E-33 INPUT AND OUTPUT FOR COMPUTER C

CARSRA, A RELIABILITY ESTIMATION TOOL FOR REDUNDANT SYSTEMS

1. 0 PROGRAM DESCRIPTION

1. 1 Introduction

CARSRA (Computer Aided Redundant System Reliability Analysis) is a FORTRAN program which is designed to facilitate the reliability assessment task for fault tolerant reconfigurable systems. It is capable of taking into account influences from transient faults and will model a wide range of redundancy management strategies.

Many previously developed reliability estimation tools are based on success paths tabulation. Examples are the TASRA program developed by Battelle Columbus Laboratories (reference 1) and the ARMM program described in reference 2. One disadvantage of the success path tabulation approach is the very large number of success paths in a highly redundant system. Another perhaps more significant disadvantage with the above mentioned programs is their inability to model transient fault and failure coverage effects where the latter parameter reflects the probability of detecting, isolating and recovering from a failure.

Recently, see for example references 3 and 4, Markov modeling has been utilized to model the effect of failure coverage. The Markov model has the advantage of offering high flexibility which makes it possible to take into account most factors of interest including fault transients, failure coverage and the possibility of spares or maintenance actions. However, a basic disadvantage with the direct Markov model approach is the very large number of Markov states required to model a system with (many) internal signal consolidation (voting) nodes.

The most significant feature of the CARSRA program is the concept of partitioning the system into smaller entities, each of which may be treated by a Markov model of lower dimension. This preserves the Markov model flexibility, avoiding the problem of an exorbitant number of Markov states.

Another significant feature of CARSRA is the ability to assess Functional Readiness as well as system failure probability. The concept of Functional Readiness is of considerable significance for a mission containing a critical subtask which will either

be performed, or not performed, depending on the operational redundancy level at the time of demand. An example is an aircraft automatic landing function for which a certain level of hardware redundancy is required before a landing may be initialized in poor visibility weather conditions.

1.2 Definition of Terms

A number of special terms are defined which will be used in the following.

A module is the smallest functional entity treated by the program. It has a Poisson type failure distribution with an a priori known failure rate.

A stage is a set of identical redundant modules. Voting or Signal Consolidation is often performed on the output signals from the modules in a stage. There are however stages without output voting. TMR stands for Triple Modular Redundancy and is associated with majority voting on the module outputs of a stage such that at least two out of three modules have to operate for the stage to survive.

A channel is a particular minimal set of modules capable of performing the system function in a non-redundant configuration.

1.3 The CARSRA Approach

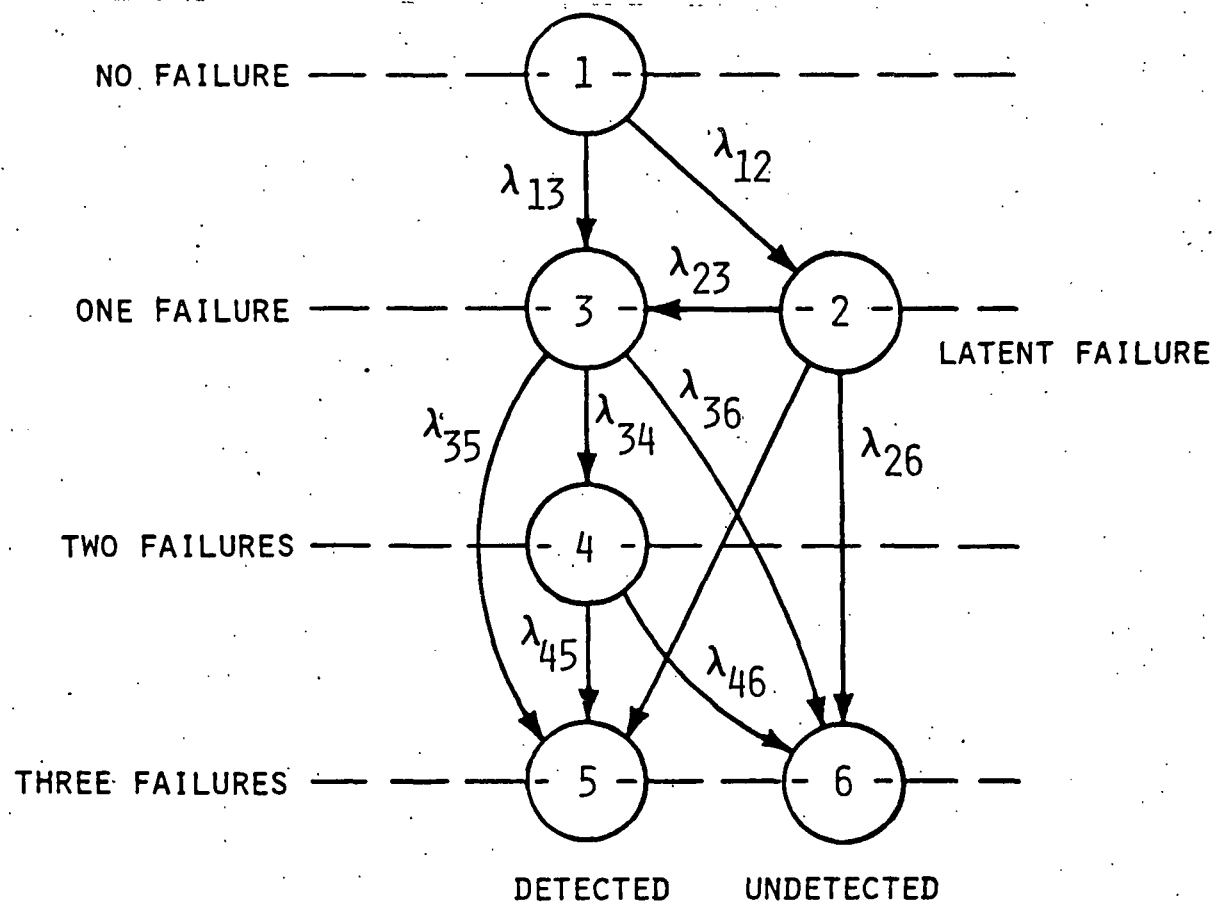
The system is conceptually partitioned into stages, for example each sensor type in a flight controls system will constitute a stage as will the processors and each servo function.

The operational status of each stage is modeled by a finite order Markov process in which each state corresponds to a particular redundancy state. An example is shown in Figure F1.

The transition rates λ_{ij} in the figure are assumed constant, i. e., not time varying and CARSRA will handle up to ten states per stage.

The operational status of a module in a particular stage may or may not depend on modules in other stages being operational. A module, which when failed will cause loss of function of another module the same channel but in a different stage, will be called a "dependency" module and the corresponding stage a "dependency stage".

FIGURE F1 EXAMPLE OF STAGE MARKOV MODEL



$$\text{EXAMPLE} = \lambda_{34} = 2\lambda_P \lambda_{34} + 2\lambda_T \ell_{34}$$

λ = PERMANENT RATE RATIO

ℓ = LEAKAGE

1.3 (cont'd)

Thus, there are two types of stages: dependency and non-dependency stages. Some stages may be both dependency and non-dependency stages. The dependency structure of a system may be described by a dependency tree diagram, an example of which is displayed in Figure F2 for a flight controls system. In this system the processor/memory stage is a dependency stage, the MPX and A/D stage are both dependency and a non-dependency stages; and each sensor stage a non-dependency stage.

The lines connecting the different stages indicate the intra-channel dependency structure in the sense that a failure in, for example, the processor channel A will cause loss of function of the channel A sensor and servo modules. The digits at the upper right hand corner of each block indicate the number of redundant modules in each stage.

The circles, to the right in the figure, represent functional elements needed for system survival. A loss of a combination of these functional elements will cause system failure. This combination may be specified by a particular entry to the program, a feature which will be further described shortly.

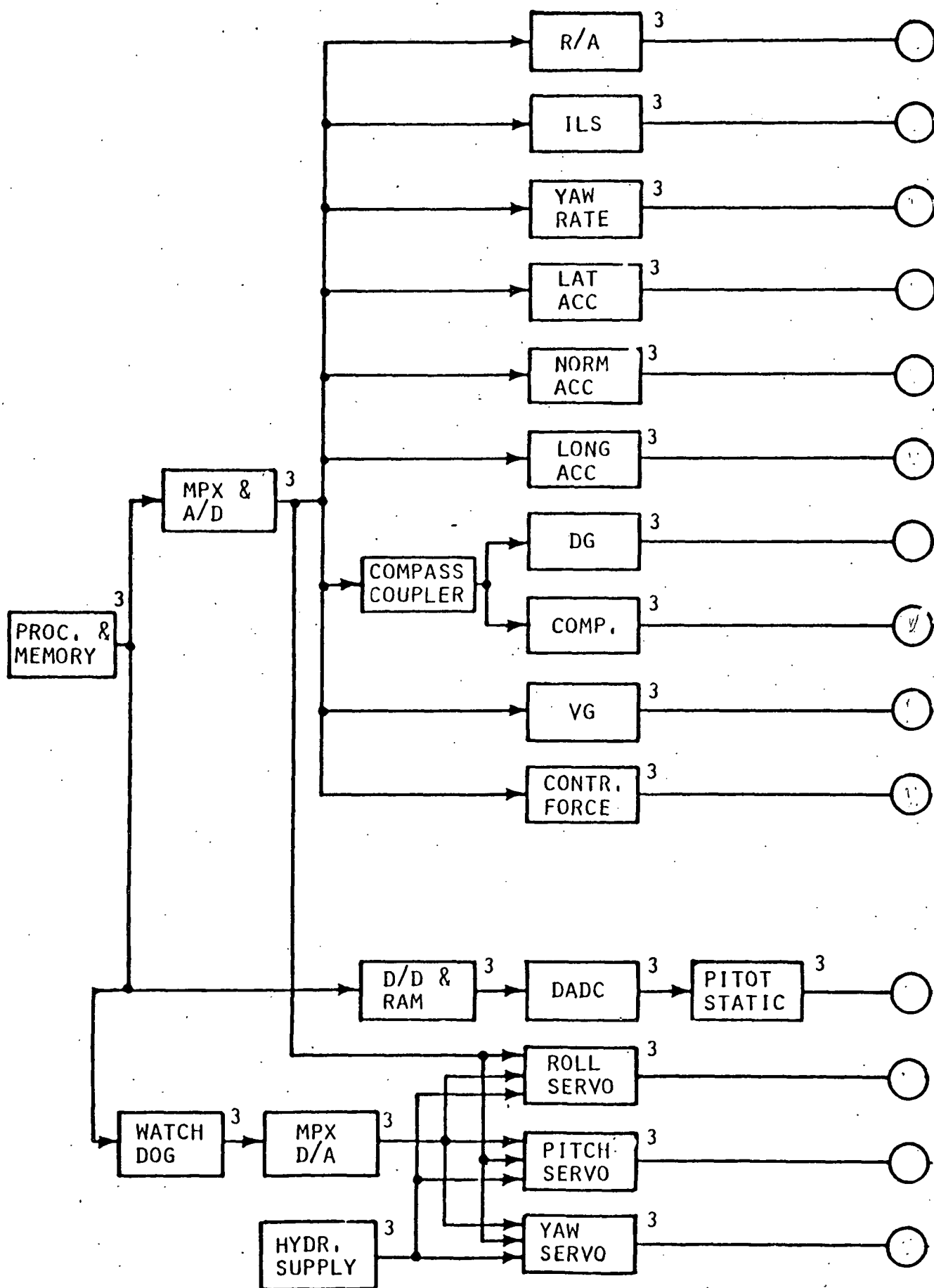
It is important to note that functional elements, which are represented by the circles, in the CARSRA model are always outputs of non-dependency stages. In a flight control system, a possible (but not unique) set of functional elements are the various sensor data, the processing function of the computer, and the actuation of the control surfaces.

CARSRA treats dependency between stages by an approach which may be denoted "exhaustive conditioning". The essence of this approach is to make the non-dependency stages independent via conditioning upon the failure status of the dependency stages.

This approach is most easily explained by presenting a simple example. Consider the triplex system outlined in Figure F3 consisting of two sensor stages A and B, a multiplex stage C and a computer stage D. The sensor signals are multiplexed and crosstrapped into the computers where signal selection (voting) and failure detection is performed in software.

TMR operation is assumed which implied that two out of three signals are required at each voting node. For simplicity the assumption is made that the output voter has zero failure rate.

FIGURE F2 FLIGHT CONTROL SYSTEM DEPENDENCY TREE



1.3 (cont'd)

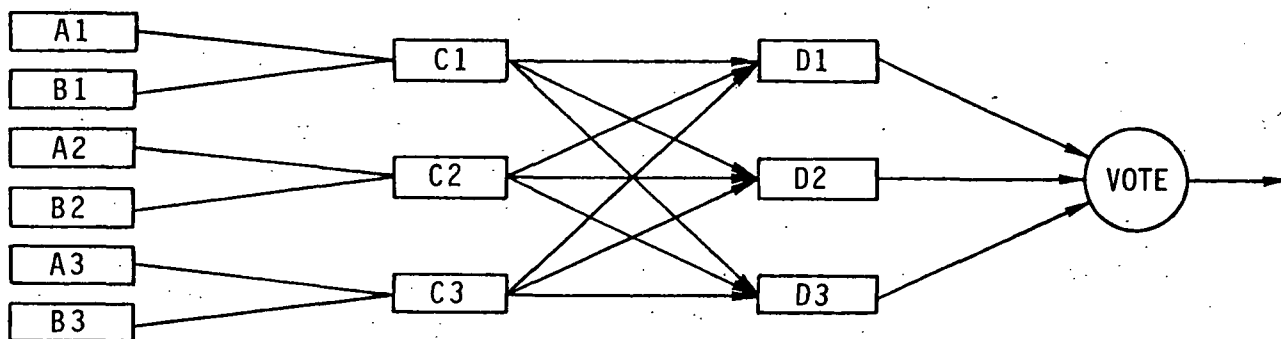
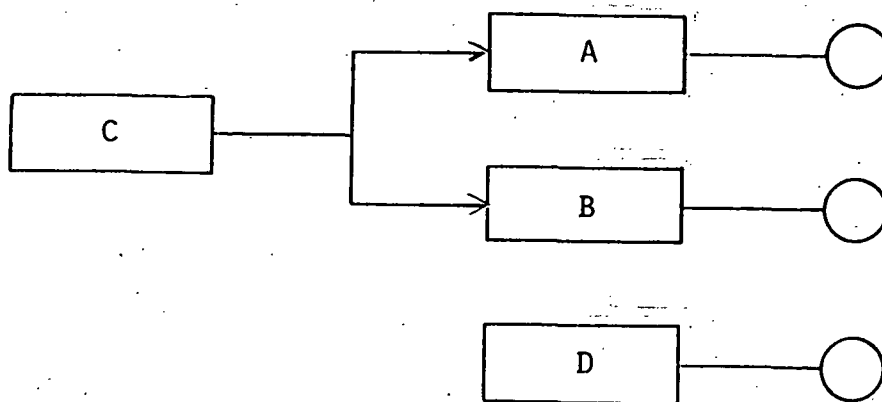


Figure F3

Note that the sensor stages and the multiplex stage are mutually dependent in the sense that a multiplex failure will cause a module failure both in sensor stage A and sensor stage B. The dependency is unidirectional since a sensor failure will not prevent the multiplex module from acquiring data from a sensor in another channel. Figure F4 displays the corresponding dependency tree.



= FUNCTIONAL ELEMENT

Figure F4

1.3 (cont'd)

To explain the approach, the following theorem will be needed:

Let E_i $i = 1, 2, \dots, n$ be disjoint events with $\sum_{i=1}^n P(E_i) = 1$.

Let F be an arbitrary event. Then,

$$P(F) = \sum_{i=1}^n P(F | E_i) \cdot P(E_i)$$

The success probability for the system of Figure F3 may now be found by defining the events E_i as follows:

- E_1 = no multiplex module failed
- E_2 = one multiplex module failed
- E_3 = two multiplex modules failed
- E_4 = all multiplex modules failed

The probability of system success may, according to the above theorem, be expanded:

$$P(S) = \sum_{i=1}^4 P(S | E_i) \cdot P(E_i)$$

The advantage of this representation is that the probabilities $P(S | E_i)$ usually are easier to find than finding $P(S)$ directly. For example:

$$P(S | E_1) = P \left\{ (\text{Stage A Survives}) \text{ and } (\text{Stage B Survives}) \text{ and } (\text{Stage D Survives}) \right\}$$

becomes with R = module reliability and $Q = 1-R$:

$$P(S | E_1) = (R_A^3 + 3R_A^2 Q_A) \cdot (R_B^3 + 3R_B^2 Q_B) \cdot (R_D^3 + 3R_D^2 Q_D)$$

Furthermore:

$$P(E_1) = R_C^3$$

1.3 (cont'd)

The next term in the expression contains the factor $P(S|E_2)$, i.e., the probability of system survival given that one multiplex module is failed. In this case both of the remaining modules in sensor stages A and B have to survive for system survival:

$$P(S|E_2) = R_A^2 \cdot R_B^2 \cdot (R_D^3 + 3R_D^2 Q_D)$$

$$P(E_2) = 3R_C^2 Q_C$$

$$\text{Finally } P(S|E_3) = P(S|E_4) = 0$$

Summarizing, the system reliability becomes:

$$\begin{aligned} P(S) = & (R_A^3 + 3R_A^2 Q_A) (R_B^3 + 3R_B^2 Q_B) (R_D^3 + 3R_D^2 Q_D) R_C^3 + \\ & + R_A^2 \cdot R_B^2 (R_D^3 + 3R_D^2 Q_D) \cdot 3R_C^2 Q_C; \end{aligned}$$

Note that this expression differs from what is obtained if the stages are assumed independent:

$$\begin{aligned} P(S_{\text{IND}}) = & (R_A^3 + 3R_A^2 Q_A) (R_B^3 + 3R_B^2 Q_B) (R_C^3 + 3R_C^2 Q_C) \cdot \\ & \cdot (R_D^3 + 3R_D^2 Q_D); \end{aligned}$$

As was mentioned above, the outputs from the non-dependency stages constitute the functional elements required for systems survival. There however, situations where these functions themselves may be redundant, for example the redundancy between aileron and spoiler control surfaces of certain aircrafts. CARSRA will model this situation by accepting a success event tabulation covering all, functional element combinations equivalent to system success.

Summarizing, the computation is performed in three different steps: Markov modeling for each stage, treating dependencies between stages via exhaustive conditioning and specifying the functions needed for success by success configuration tabulation.

1.4 The Functional Readiness Feature

Fault tolerant systems will continue to perform their functions even after experiencing one or several module failures. With this basic feature it will be of interest to be able to assess the probability of experiencing a certain redundancy degradation within a prescribed time interval and furthermore to be able to assess the system

failure probability given that a certain degradation has taken place. This information could be used to establish Functional Readiness Criteria, i.e., the system failure states that will not cause deferment of a particularly critical phase of a mission, for example the landing of a manned spacecraft on the moon or the previously mentioned case of automatic landing of a passenger transport in low visibility weather conditions.

CARSRA accepts a selected Functional Readiness Criterion specifying the combination of modules which could be failed and computes the probability of having any of these modules failed as a function of time. It also computes the system failure given a Functional Readiness Criterion as a function of time in separately specified time frame. Two different system failure modes may be specified, e.g., detected or undetected system failure.

The following relations are used. Let $PFR(t_1)$, $i=1, 2, \dots, N$ denote the probabilities associated with the different specified Functional Readiness configurations at a time t_1 and let $PPF_i(t_2)$ be the conditional probability of a certain failure mode at an exposure time t_2 given Functional Readiness configuration i :

CARSRA then computes:

$$PFR = \text{Functional Readiness} = \sum_{i=1}^N PFR_i(t_1)$$

$$PPF = \text{Failure Probability} = \left[\sum_{i=1}^N PPF_i(t_2) \times PFR_i(t_1) \right] \cdot PFR^{-1}$$

In addition to the above mentioned features, two levels of computational accuracy may be specified. The resulting computational roundoff errors are indicated in the computer printout.

1.5 CARSRA Program Structure

In this section, the structure of the program will be described in enough detail to provide a basic understanding of the operation.

The program is designed using a top-down approach with each subtask carried out in a separate subroutine. The overall structure is shown in Figure F5.

MAIN is the main program and each of the other blocks represents subroutines with a higher order subroutine calling a lower order routine as indicated by the lines connecting the different blocks. The subroutine FAILPR will, for example, call the subroutine SETETY, STRIP, EQUAL, PRINTZ, INDFP and DEFPF.

The operation of each of the subroutines and their interplay will next be described.

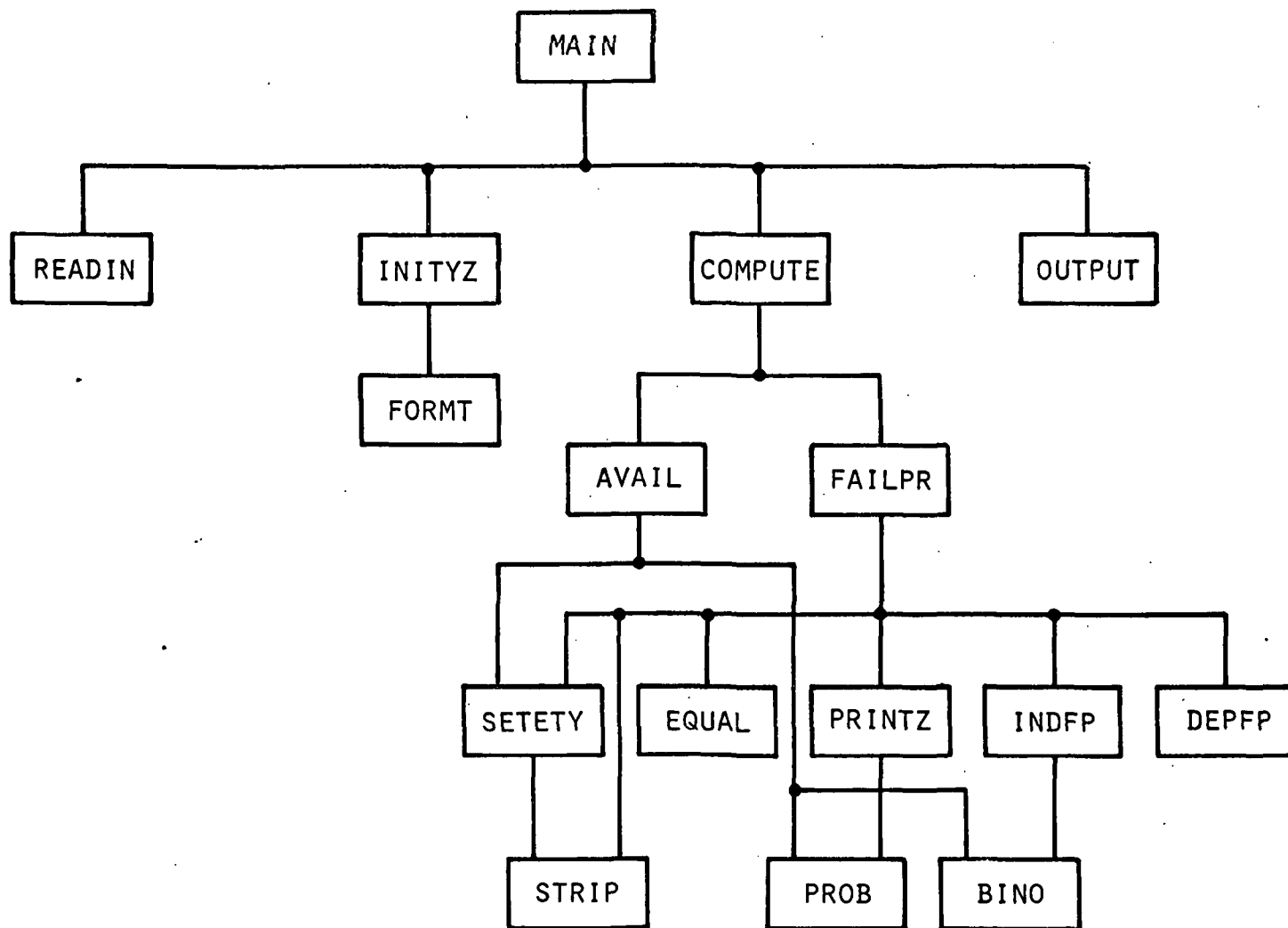


FIGURE F5 CARsRA STRUCTURE

405
405

1.5.1 Subroutine Descriptions and Flow Diagrams

MAIN: The MAIN program directs the data input, the computation, and the data printout. It calls the subroutine READIN, INITYZ, COMPUTE and OUTPUT. As part of the input data from READIN, the MAIN program gets the specified Functional Readiness time interval and time increment which it uses to set up a loop which computes Functional Readiness and system Failure Probability data and outputs this information for each Functional Readiness time increment. The MAIN flow diagram is shown in Figure F6.

READIN: Reads input data from a punched card file specifying Markov model transition rates, desired Functional Readiness time interval and increment, desired Failure Probability time interval and increment; Functional Readiness Criteria, the Success Configuration and the desired computational accuracy. The subroutine flow diagram is indicated in Figure F7.

INITYZ: Initializes the computation by computing the transition rates

$$\lambda_{ii} = - \sum_j \lambda_{ij}$$

It calls the subroutine FORMT which computes a matrix T and its inverse TINV used by the program to solve the Markov model equations. (The mathematical details

may be found at the end of this appendix.) The subroutine INITYZ also initializes the indicator array INDIC which is used to indicate the Functional Readiness Configuration, and the array INTMP which is used to map the dependency between modules. Also, the array MAP is constructed indicating the relation between modules failed in a stage and the number of the corresponding Markov state. The flow diagram is shown in Figure F8.

FORMT (I):

Computes the matrices T and TINV for a particular stage (see solution at the end of this appendix). I indicates the stage number. The flow diagram is displayed in Figure F9.

COMPUTE (AVT, AVBTY):

Computes the detected and undetected failure (or any other two selected failure mode) probabilities for a certain Functional Readiness time and criterion. The failure probabilities are stored in common arrays FAILP and FALUN where FAILP is the

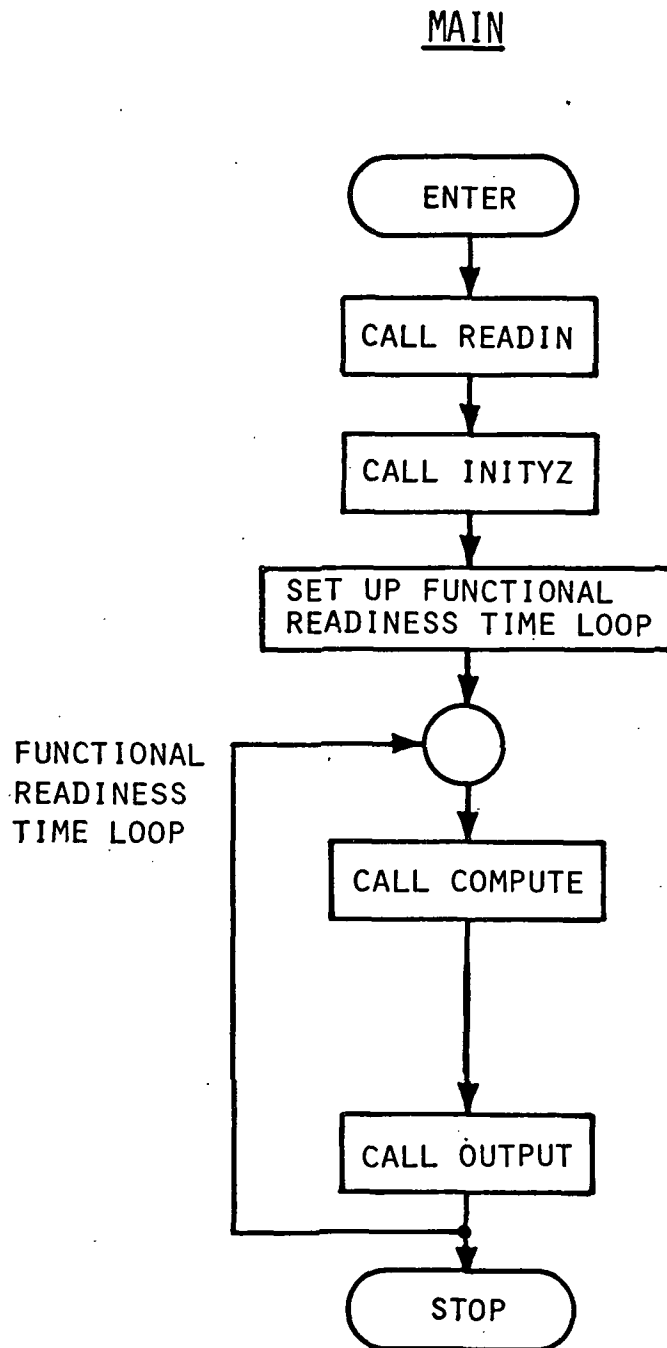


FIGURE F6 MAIN PROGRAM FLOW DIAGRAM

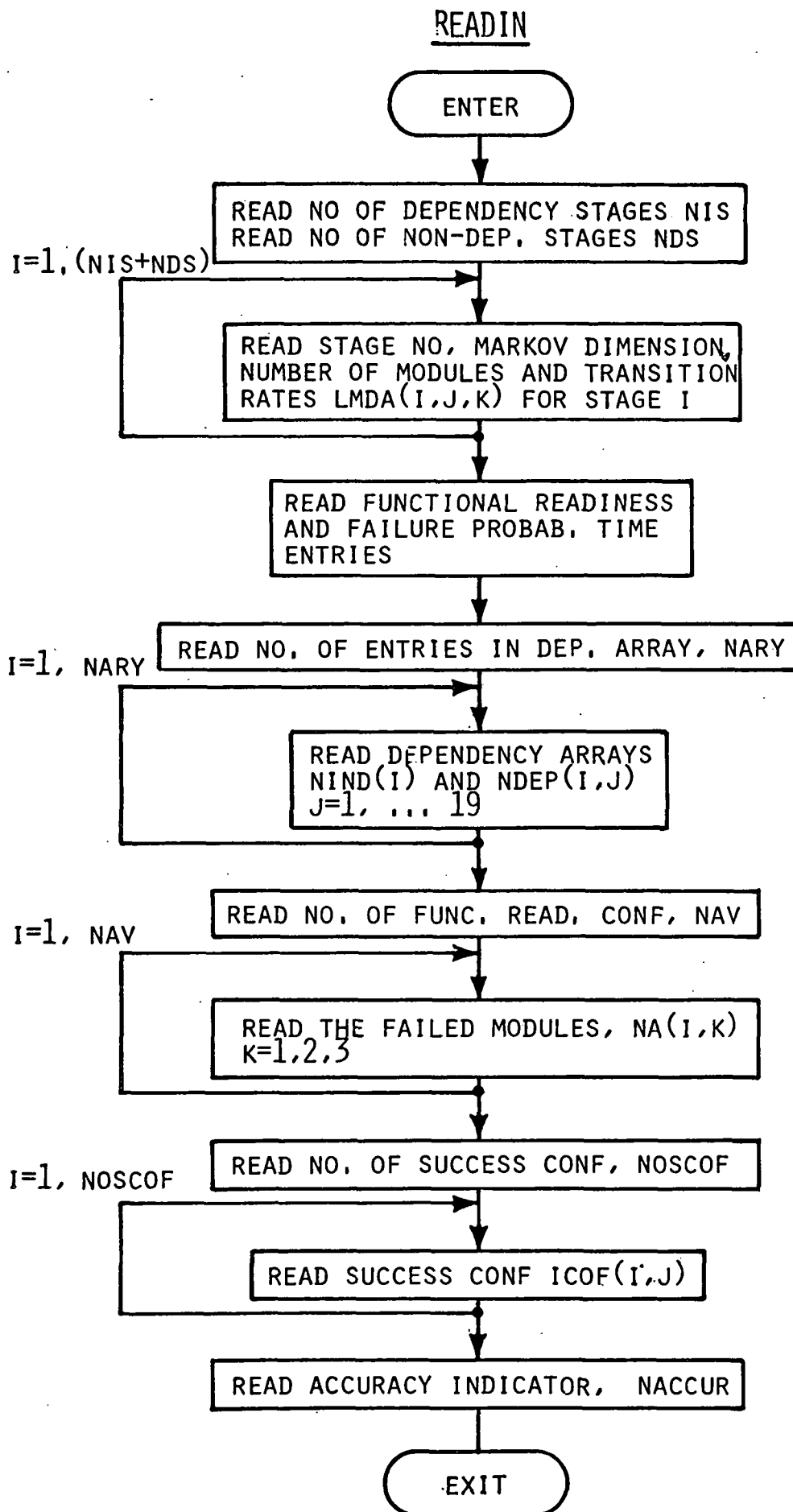


FIGURE F7 READIN FLOW DIAGRAM

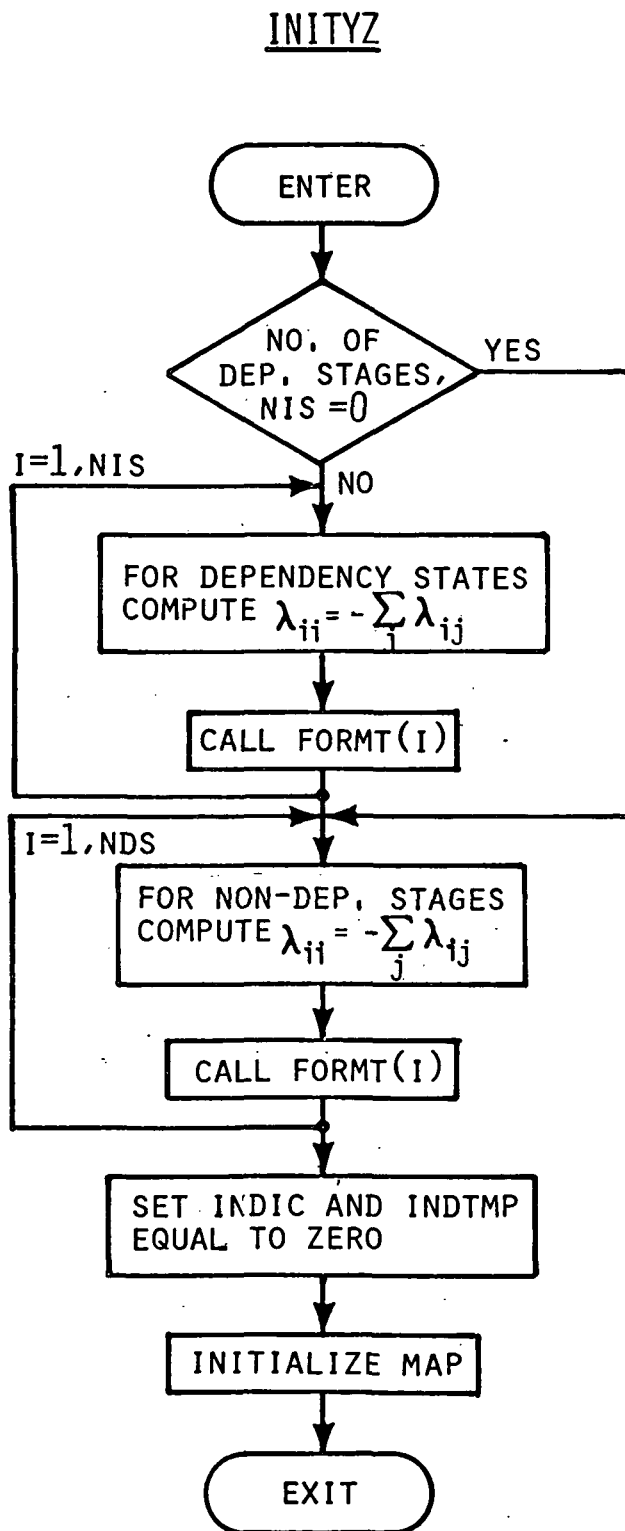


FIGURE F8 INITYZ FLOW DIAGRAM

FORMT

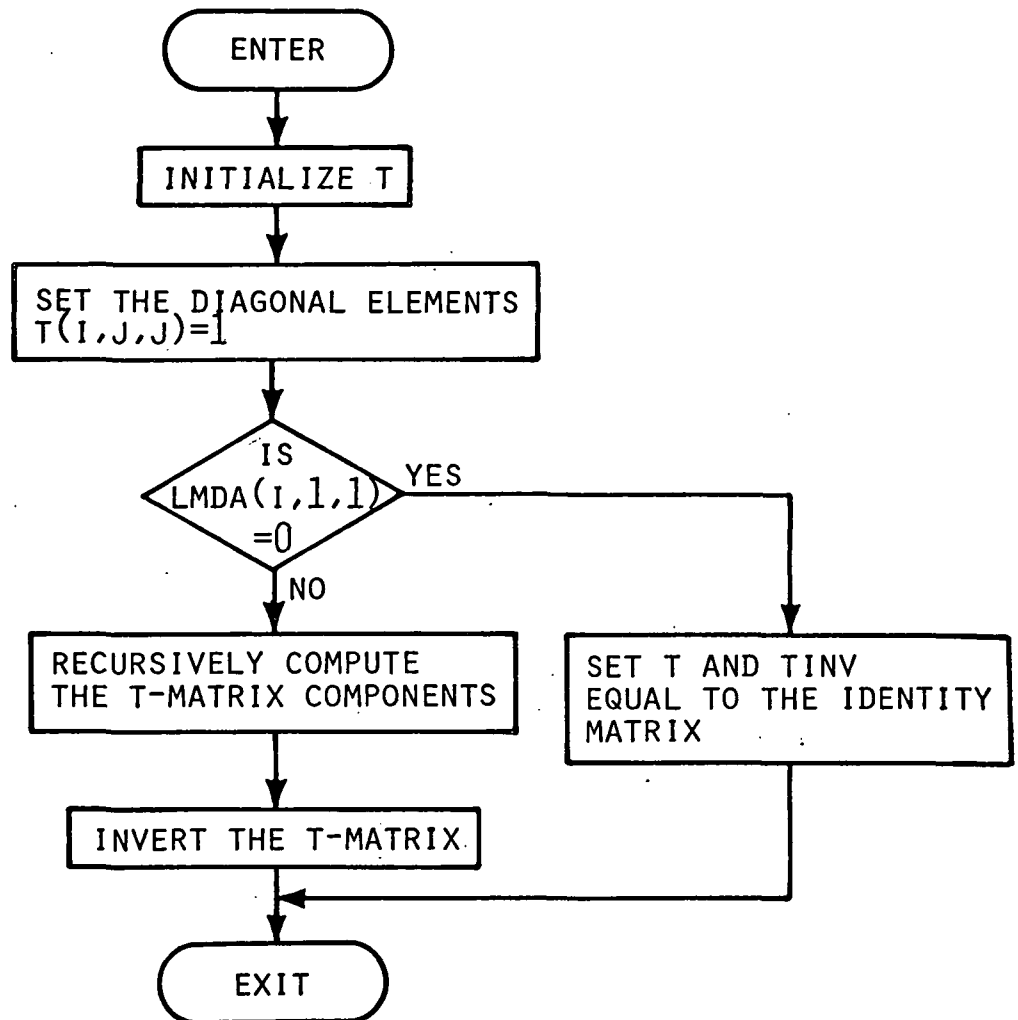


FIGURE F9 FORMT FLOW DIAGRAM

1.5.1 (cont'd)

probability of being in any of the two last Markov states in any stage and FALUN is the probability of being in the last Markov state in any stage. The array SPY which contains the computational truncation error estimates is also computed. The range of the above arrays are specified by the failure probability time range FPMT and the increment FPDT read by INITYZ, with the i element in each array corresponding to a failure probability time $T = i \cdot \text{FPDT}$. The Functional Readiness time is transferred from MAIN to COMPUTE in the argument AVT. COMPUTE calls the subroutine AVAIL to compute the Functional Readiness probability which is transferred to MAIN in the argument ABVTY. The flow diagram is shown in Figure F10.

OUTPUT (AVT, AVBTY):

Outputs the arrays FAILP, FALUN and SPY together with the Functional Readiness (AVBTY) for a particular Functional Readiness time (AVT).

AVAIL (PRO, I, TIME):

Computes the Functional Readiness, PRO, at time TIME for the Functional Readiness configuration specified by entry number I in the Functional Readiness table which is read by READIN and stored in the common array NA (I, K). The AVAIL subroutine also sets the indicator array INDIC corresponding to the Functional Readiness Configuration and, in the case the Functional Readiness configuration specifies a dependency module failure, rearranges the structure of the common arrays NIND and NDEP which specifies module dependencies. The corresponding rearranged arrays are NTIND and NTDEP.

The subroutine AVAIL uses the subroutine PROB to compute Functional Readiness and SETETY to set the indicator array INDIC. The flow diagram is shown in Figure F11.

COMPUTE

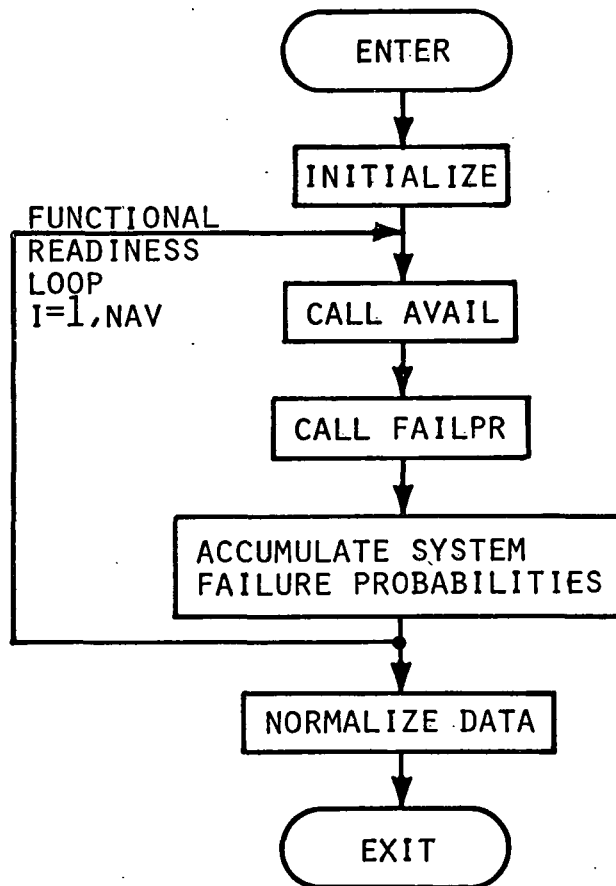


FIGURE F1Q COMPUTE FLOW DIAGRAM

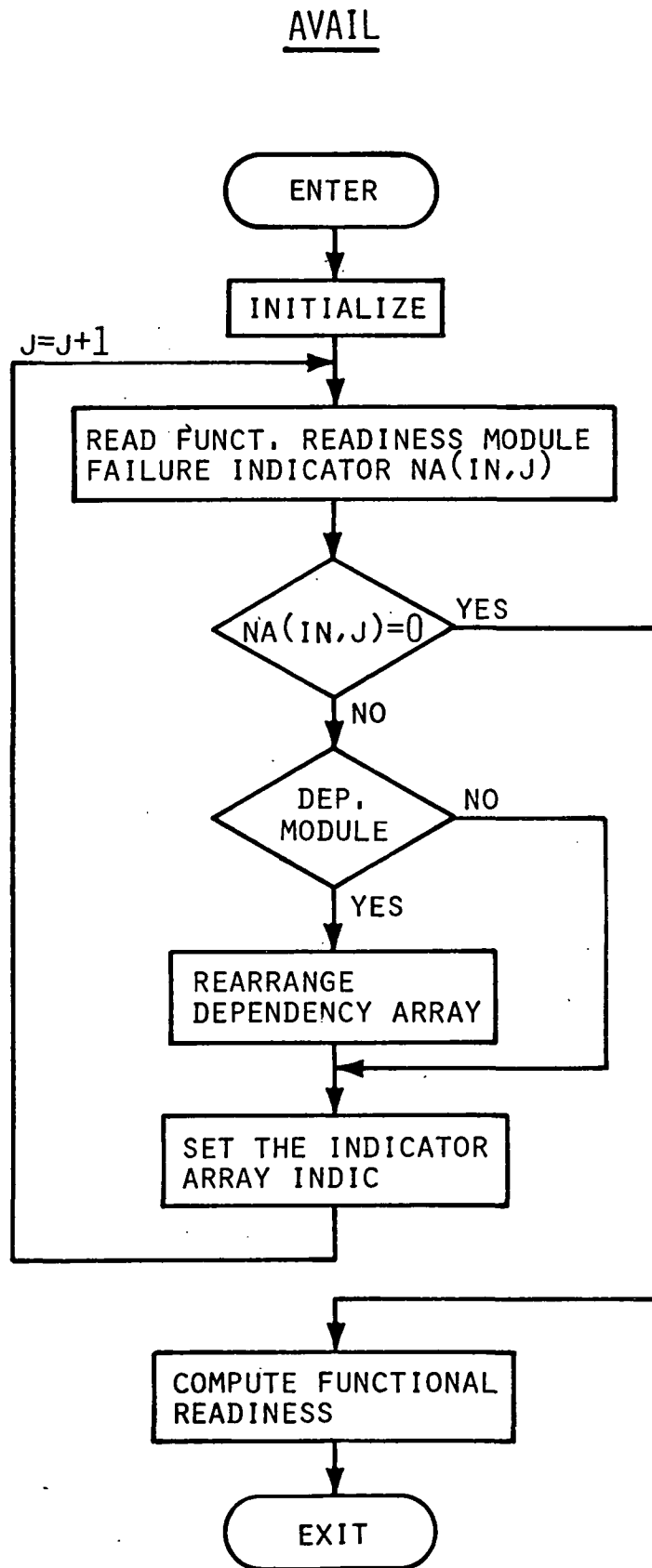


FIGURE F11 SUBROUTINE AVAIL FLOW DIAGRAM

PROB (ISTAGE, IENTRY, IEXIT, P, TIME):

This subroutine computes for stage ISTAGE, the probability P of being in Markov state IEXIT at time TIME given state IENTRY at time zero. The subroutine uses the previously, in FORMT, calculated matrices T and TINV.

SETETY (J):

The subroutine sets the failure condition array INDTMP according to the module failure pattern specified by entry number J in the dependency table which specified the system dependency structure. The flow diagram is shown in Figure F12.

STRIP (INPUT, NSTGE, NSTATE):

Finds the stage number XX and state Y from a three digit number INPUT = XXY.

FAILPR (FALP, FALND, SP):

This subroutine computes the failure probability arrays FAILP and FALUN for a certain Functional Readiness configuration specified by the status of the array INDIC which was set in AVAIL. It uses the (rearranged) dependency arrays NTIND (I) scanning through all entries I and failing combinations of dependency modules NTIND (I) which in turn causes non-dependency modules NTDEP (I, J), J = 1, 2, ... to fail. This is the actual implementation of the above described "exhaustive conditioning" approach. The probability of each combination of dependency module failure states is computed by calling the subroutine INDFP, and the conditional probability of the non-dependency modules failing, given the particular combination of dependency module failures, is computed by calling DEPFP. The system failure probability is then computed by multiplying these two probabilities and summing over the different dependency module failure combinations. To avoid calling the subroutine PROB repeatedly all needed transition probabilities are computed initially by calling PRINTZ which stores those probabilities in the array PROBAB which is common for INDFP and DEPFP.

Combinations of up to two different dependency module failures are considered if the accuracy indicator NACCUR entered by READIN is set to zero. With NACCUR = 1, combinations of up to three dependency module failures will be considered.

SETETY(J)

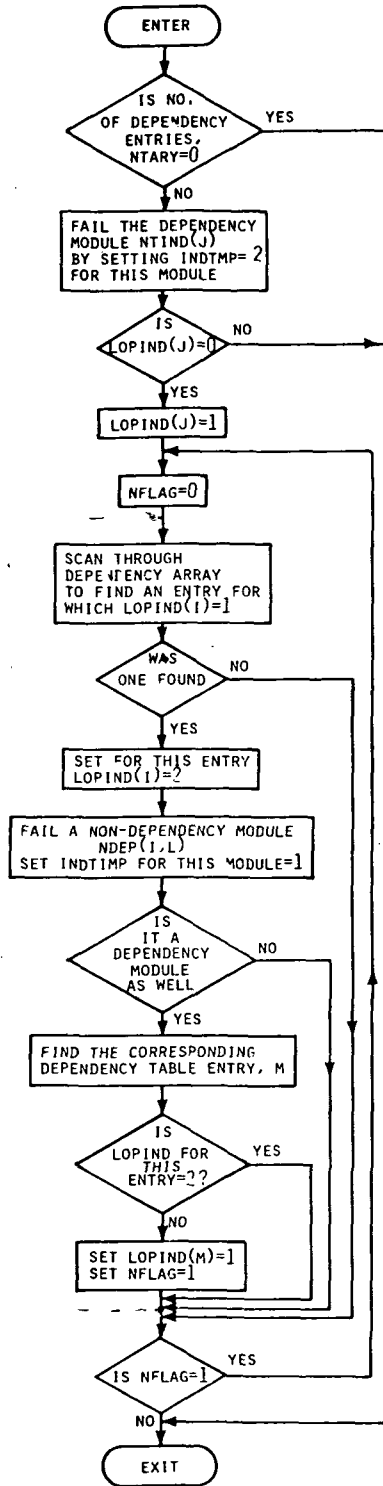


FIGURE F12 SUBROUTINE SETETY FLOW

1.5.1 (cont'd)

The truncation error, caused by not covering of possible combinations of dependency module failures, is the difference between unity and the sum of the probabilities of all considered dependency module combinations. A flow diagram over the subroutine FAILPR is outlined in Figure F13.

PRINTZ (TIME, NIS, NDS, DIM):

Computes Markov transition probabilities from state K to state J with $J \geq K$ for all system stages and stores the result in the array PROBAB.

EQUAL (A, B):

Equalizes the two dimensional array A with the two dimensional array B.

INDFP (PR, FP, FPU, K):

Computes the probability PR of a certain dependency module failure pattern specified by the state of the array INDTMP. The possibility that this specified combination leads to system failure is also computed and the two failure mode probabilities are stored in FP and FPU. The flow diagram is in Figure F14.

BINO (M, N, K):

Computes the binomial coefficient $K = \binom{M}{N}$.

DEFP (PFAIL, PUNDET):

Computes the conditional probabilities of two system failure modes given a particular dependency module failure state which causes failure of non-dependency modules as specified by the status of the array INDTMP. It also scans through the stage success table and accumulates success probabilities over all combinations of functional elements equivalent to system success. The flow diagram may be found in Figure F15.

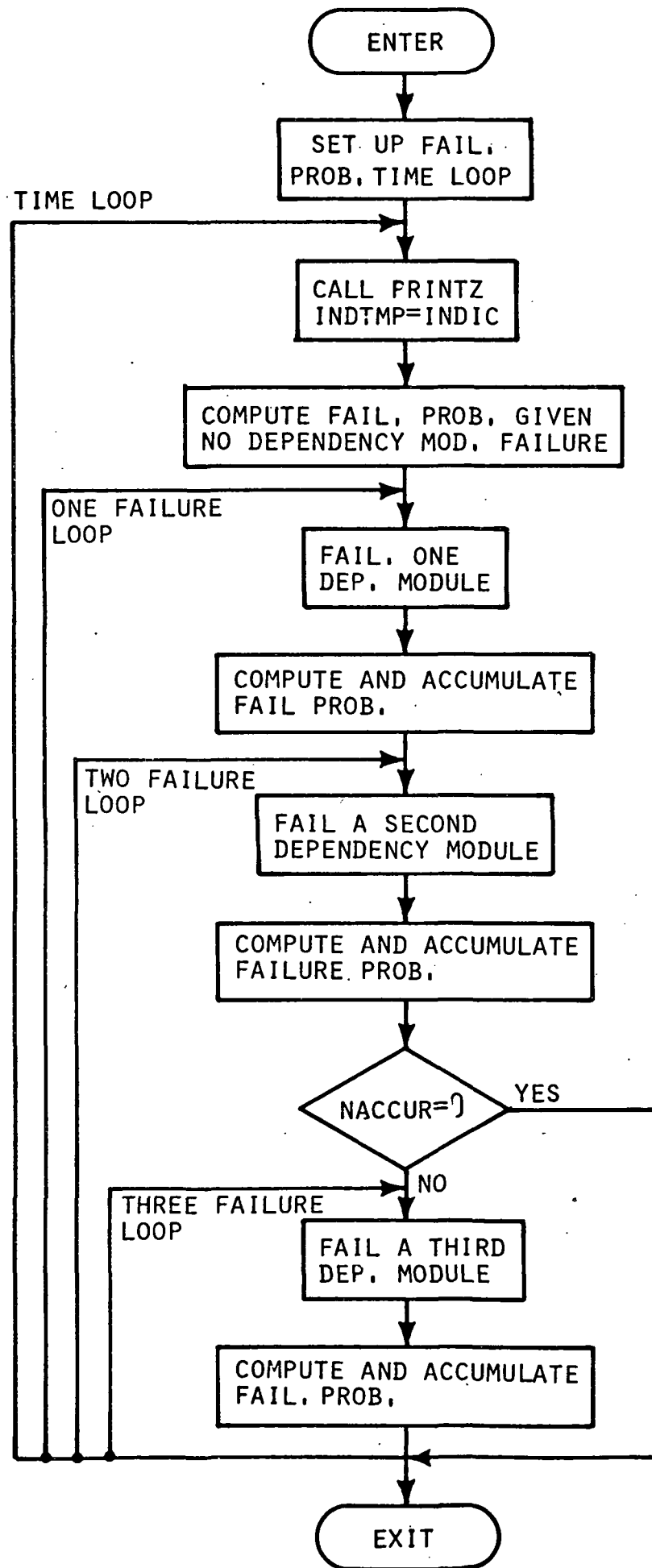


FIGURE F13
FLOW DIAGRAM FOR
SUBROUTINE FAILPR

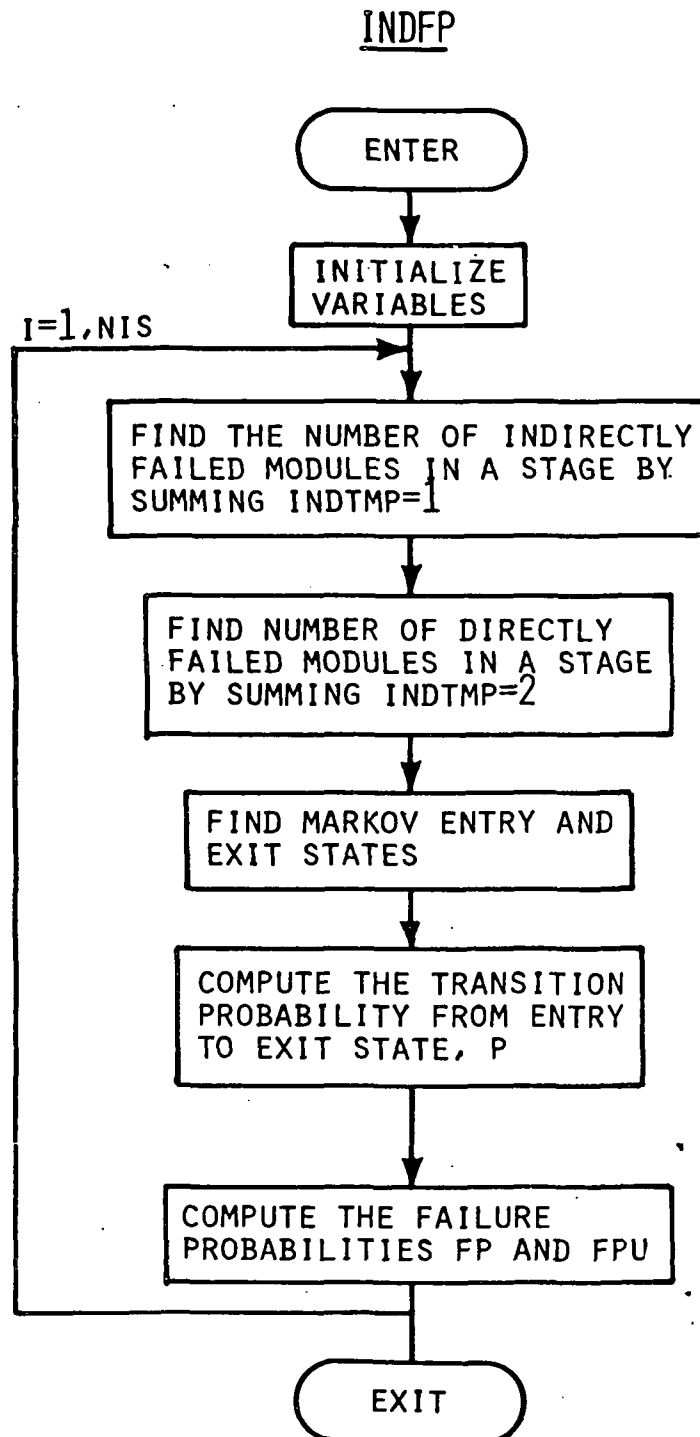


FIGURE F14 INDFP FLOW DIAGRAM

DEPFP

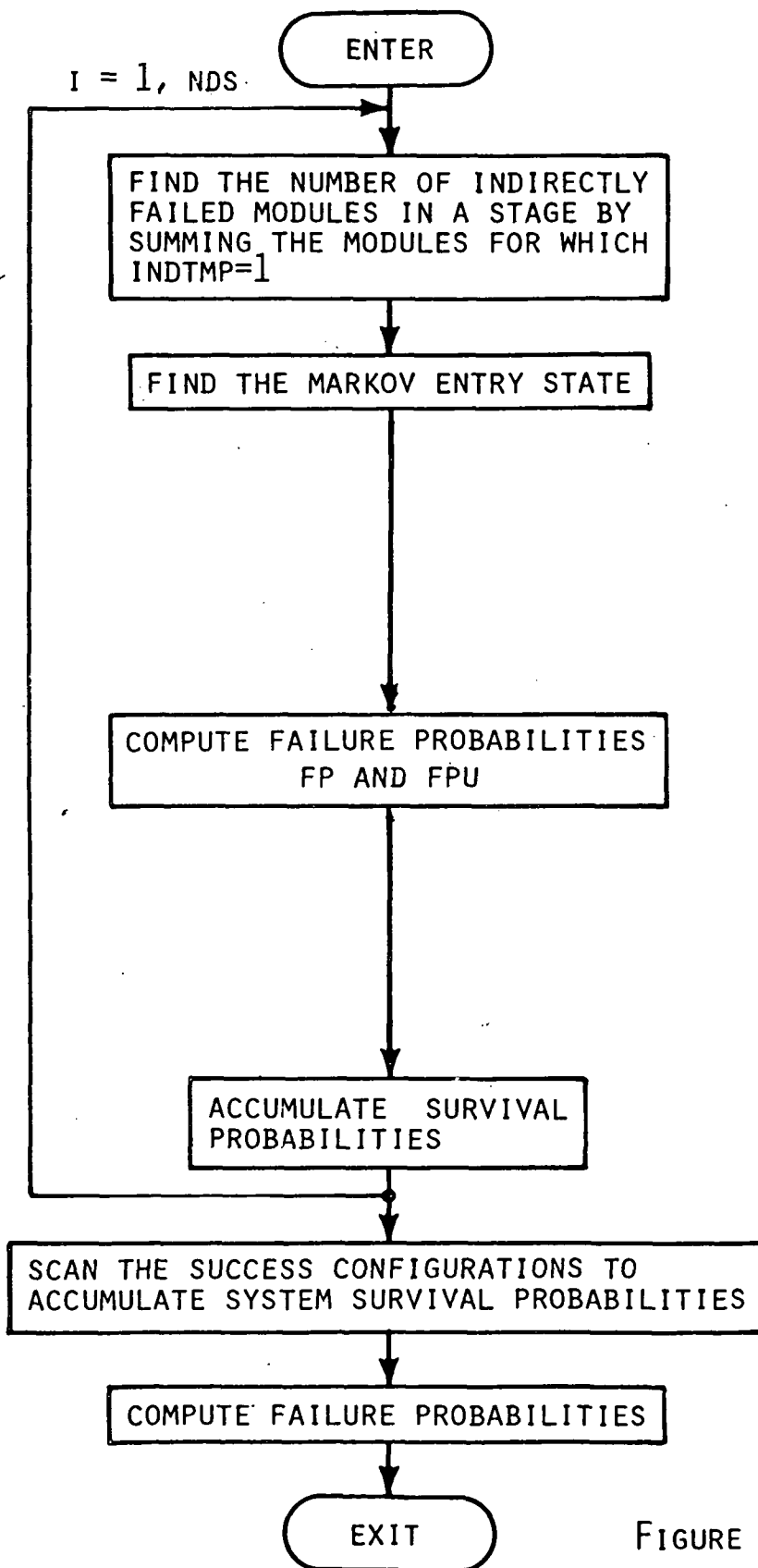


FIGURE F15 DEPFP FLOW DIAGRAM

2.0 CARSRA — A USER'S GUIDE

2.1 General

The CARSRA program is coded in FORTRAN IV with approximately 700 FORTRAN statements. It requires 100,000 core locations to run, and the execution time varies with the complexity of the system to be analyzed and the selected program option, the typical execution time being in the range of 1-30 seconds.

2.2 Input Description

All input data is on punched cards and must be input in a prescribed order. The following comments are made in relation to

Table F1	which	specifies
----------	-------	-----------

 required input data, the corresponding variable names used by the program, and the input format.

2.2.1 Dependency and Non-Dependency Stages

The system is partitioned into dependency and non-dependency stages with a failure of a dependency stage module causing failure of a module in a different stage. In cases where a stage both is a dependency and a non-dependency stage, it will, in the program input, be identified as a dependency stage.

Dependency stages are assigned numbers in the range 1-20 consecutively starting by 1. If failure of a certain dependency stage module causes another dependency stage module to fail, the former stage should be assigned a lower stage number than the latter. Dependency stage numbers should be assigned in consecutive order without leaving a number unassigned inside the array.

Non-dependency stages are assigned numbers in the range 21-50 consecutively starting with 21.

2.2.2 Stage Dimension, Numbers of Modules and Transition Rates

The assigned stage number (NST), the dimension (NDIM) and the number of modules in the stage (MODN) information is entered on one card. The dimension specifies the number of states in the Markov model for the stage and is used to control the read in of the following (NDIM-1) cards which

TABLE F1: CARSRA Input Data

Input Data	Variable Name	Format	No. of Cards
No. of non-dependency (NDS) and dependency stages (NIS)	NDS, NIS	2 I 10	1
Stage dimension and No. of modules for a certain stage, NST, followed by	NST, NDIM, MODN	3 I 10	1 } NDS+NIS TIMES
The corresponding transition rate matrix in failures per million hours.	LMDA (NST, K, J) J = 1, NDIM K = 1, (NDIM-1)	10(F8.2)	
Functional readiness and failure probability time entries	AMT, ADT, FPMT, FPDT	4(F10.5)	1
No. of dependency array entries	NARY	I 10	1
Dependency Structure	NIND(I), NDEP(I, J) I = 1, NARY J = 1, 19	20(I4)	NARY
No. of Functional Readiness Configurations	NAV	I 10	1
Failed Modules	NA (I, K) I = 1, NAV K = 1, 3	3(I4)	NAV
No. of Success Configurations	NOSCOF	I 10	1
Success Configurations	ICOF (I, J) I = NOSCOF J = 1, 50	50 I 1	NOSCOF
Accuracy Indicator	NACCUR	I 10	1

2.2.2 (cont'd)

specify the transition rates LMDA (NST, K, J) in failures per million hours. LMDA (NST, K, J) is the transition from state K to state J in stage NST with LMDA (NST, K, J), $J = 1, 2, \dots, \text{NDIM}$ on one card for each K value. Only (NDIM-1) cards corresponding to $K = 1, 2, \dots, (\text{NDIM}-1)$ have to be entered for each stage since the last state always will have zero transition rates. Only transitions from lower order to a higher order states are permitted, i.e., LMDA (NST, K, J) must be equal to zero (left blank) for $K > J$. This constraint implies that CARSRA as currently coded is unable to handle modeling of equipment repair.

Markov state one always models no module failures, Markov state two one module failure, and state three two module failures.

2.2.3 Functional Readiness and Failure Probability Time Entries

Transitional Readiness Time span (AMT) and time increment (ADT) are entered on one card together with Failure Probability Time span (FPMT) and time increment (FPDT). For each Functional Readiness Time equal to $I \times \text{ADT} \leq \text{AMT}$, $I = 0, 1, 2, \dots$, a table over the Failure Probabilities as a function of Failure Probability Time of $J \times \text{FPDT} < \text{FPM}$ $J = 1, 2, \dots$ is printed. If only the Failure Probabilities are of interest, enter $\text{AMT} = 0$, $\text{ADT} = 1$.

2.2.4 Dependency Array

The number of dependency modules in the system, (NARY), is entered on a separate card, followed by NARY cards specifying the system dependency configuration. Each dependency module NIND (I) ($I = 1, \dots, \text{NARY}$) will when failed cause failure of modules NDEP (I, J), $J = 1, \dots, N$ with $N \leq 19$. The modules are specified by NIND and NDEP in the form XXY with XX being the stage number and Y the module number within the stage. (See further the example below).

2.2.5 Functional Readiness Table

The number of Functional Readiness configuration entries NAV is specified on a separate card followed by NAV cards, one for each configuration. Each configuration is characterized by up to 3 failed modules NA (I, K) K = 1, 2, 3 where the module is indicated by XXY as before (2.2.4). The Functional Readiness probability computed by the program is the probability of having any one of the specified system failure patterns at a given time.

2.2.6 Stage Success Table

NOSCOF, entered on a separate card, specifies the number of stage failure patterns equivalent to system success. One card is thereafter entered for each pattern which is specified by a 1 (one) in a column corresponding to a failed stage. If all stages are essential for system success, NOSCOF is equal to one followed by a blank card. If two stages, for example the non-dependency stages 21 and 22, are redundant three cards are required:

1) Col 21 = 0 Col 22 = 0; 2) Col 21 = 1 Col 22 = 0; 3) Col 21 = 0 Col 22 = 1.

2.2.7 Accuracy Indicator

The accuracy indicator, NACCUR, specifies the level to which conditioning upon the combinations of dependency module failure is performed. If NACCUR = 0 (lower accuracy setting) all combinations of zero, one and two dependency module failures will be considered plus failure combinations equivalent to system failure. If NACCUR = 1, up to three dependency module failures will be considered plus failure combinations equivalent to system failure. The higher accuracy could be required when treating a system with high module redundancy (for example a quad system) or when the mission time is long. However, the program run time could be in the order of ten times longer for the higher accuracy setting.

The truncation error is indicated by an accuracy number. The correct failure probability value will lie in the range

[printed output value, printed output value + accuracy] .

2.3

Output Description

A table over system failure probabilities are printed for each Functional Readiness Time $I \times ADT = AMT$. The table entries are Failure Probability times $J \times FPDT < FPMT$, the probabilities of a detected and an undetected system failure (or any other two failure modes), and the truncation error band.

2.4

An Example

The use of the CARSRA program will be illustrated by a simple example. Consider a system with dependency structure depicted in Figure F16.

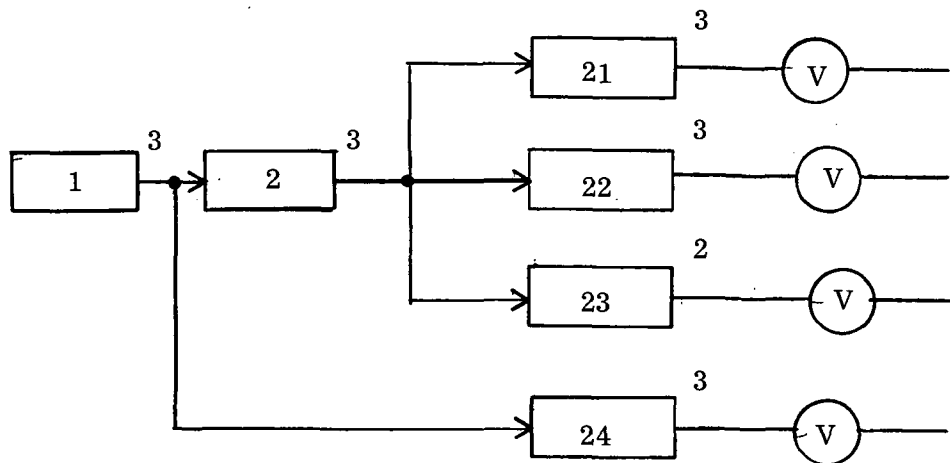


Figure F16

There are two dependency stages, 1 and 2, and four non-dependency stages, 21-24. The numbers at the right hand upper corners of the stage blocks pertain to the modular redundancy level.

Assume further that the stage Markov model is the one depicted in Figure F17 for the five triple redundant stages and like in Figure F18 for the single duplex stage.

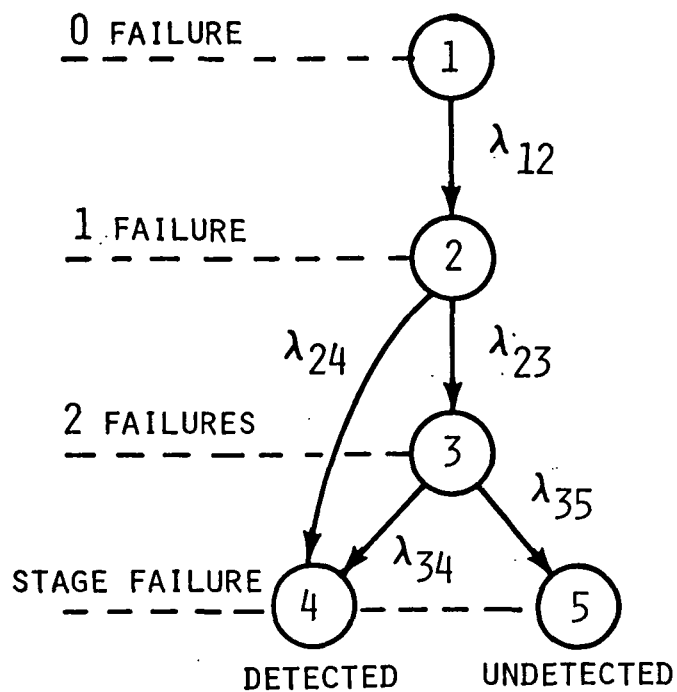


FIGURE F17 TRIPLEX STAGE MARKOV MODEL

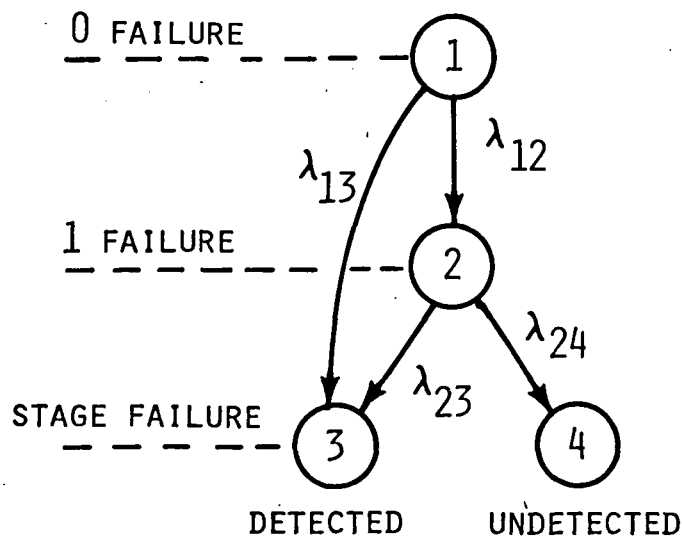


FIGURE F18 DUPLEX STAGE MARKOV MODEL

The transition rates are given by Table F2:

TABLE F2. CARSRA Example Transition Rates/ 10^6 Hrs

Stage	λ_{12}	λ_{13}	λ_{23}	λ_{24}	λ_{34}	λ_{35}
1	300	0	180	20	90	10
2	60	0	32	8	15	5
21	900	0	600	0	300	0
22	750	0	450	50	200	50
23	180	20	50	50	—	—
24	300	0	150	50	50	50

Functional Readiness data is desired in 50-hour increments (ADT) up to 100 hours (AMT) for a Functional Readiness criterion corresponding to a single module failure in any of the stages 21, 22 or 24.

Failure probabilities in 1-hour increments (FPDT) up to 5 hours (FPMT) are to be generated.

Stages 21 and 22 are redundant in the sense that the system will survive a loss of any one but not both of these stage functions.

The CARSRA input data deck is presented in Table F3.

The output is presented in Table F4. The input data is first echoed after which the computed reliability data is listed. The negative accuracy band is caused by computational roundoff errors.

TABLE F4: CARSRA Output Example

ECHO CHECK

NO OF DEP STAGES= 4 NO OF IND STAGES= 2

STAGE 1 DIMENSION 5 NO OF MODULES 3

-0.	300.	-0.	-0.	-0.
-0.	-0.	180.	20.	-0.
-0.	-0.	-0.	90.	10.
-0.	-0.	-0.	-0.	-0.

STAGE 2 DIMENSION 5 NO OF MODULES 3

-0.	60.	-0.	-0.	-0.
-0.	-0.	32.	8.	-0.
-0.	-0.	-0.	15.	5.
-0.	-0.	-0.	-0.	-0.

STAGE 21 DIMENSION 5 NO OF MODULES 3

-0.	900.	-0.	-0.	-0.
-0.	-0.	600.	-0.	-0.
-0.	-0.	-0.	300.	-0.
-0.	-0.	-0.	-0.	-0.

STAGE 22 DIMENSION 5 NO OF MODULES 3

-0.	750.	-0.	-0.	-0.
-0.	-0.	450.	50.	-0.
-0.	-0.	-0.	200.	50.
-0.	-0.	-0.	-0.	-0.

STAGE 23 DIMENSION 4 NO OF MODULES 2

-0.	180.	20.	-0.
-0.	-0.	50.	50.
-0.	-0.	-0.	-0.

STAGE 24 DIMENSION 5 NO OF MODULES 3

TABLE F4: (cont'd)

-0.	300.	-0.	-0.	-0.
-0.	-0.	150.	50.	-0.
-0.	-0.	-0.	50.	50.
-0.	-0.	-0.	-0.	-0.

AMT= 100.00000 ADT= 50.00000 FPM= 5.00000 FPDT= 1.00000

DEPENDENCY ARRAYS

11	21	241	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0
12	22	242	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0
13	23	243	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0
21	211	221	231	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0
22	212	222	232	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0
23	213	223	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0

NUM OF AVAILABILITY CONFIG 10

-0	-0	-0
211	-0	-0
212	-0	-0
213	-0	-0
221	-0	-0
222	-0	-0
223	-0	-0
241	-0	-0
242	-0	-0
243	-0	-0

NUM OF SUCCESS CONFIG 3

ACCURACY, EQUAL TO ZERO, IF LOWER 0

TABLE F4: (cont'd)

COMPUTED RELIABILITY DATAS

AVAILABILITY AT TIME 0.0000 HOURS .100000E+01

SYSTEM FAILURE PROBABILITIES

EXPOSURE TIME	FAIL. PROB.	UNDET. FAIL PROB.	ACCURACY
1.000000	.2003660E-04	.1057351E-07	-.5634342E-13
2.000000	.4014657E-04	.4202806E-07	-.1421085E-13
3.000000	.6032923E-04	.9459467E-07	-.3552714E-13
4.000000	.8053512E-04	.1582242E-06	-.4263256E-13
5.000000	.1009140E-03	.2629374E-06	-.4263256E-13

AVAILABILITY AT TIME 50.0000 HOURS .908596E+00

SYSTEM FAILURE PROBABILITIES

EXPOSURE TIME	FAIL. PROB.	UNDET. FAIL PROB.	ACCURACY
1.000000	.2072109E-04	.1121557E-07	-.5684342E-13
2.000000	.4151492E-04	.4487415E-07	-.1421085E-13
3.000000	.6238142E-04	.1009935E-06	-.3552714E-13
4.000000	.8332054E-04	.1795913E-06	-.4263256E-13
5.000000	.1043322E-03	.2806852E-06	-.4263256E-13

AVAILABILITY AT TIME 100.0000 HOURS .931650E+00

SYSTEM FAILURE PROBABILITIES

EXPOSURE TIME	FAIL. PROB.	UNDET. FAIL PROB.	ACCURACY
1.000000	.2129515E-04	.1181686E-07	-.5684342E-13
2.000000	.4266268E-04	.4727749E-07	-.1421085E-13
3.000000	.6410253E-04	.1063970E-06	-.3552714E-13

TABLE F4: (cont'd)

4.000000 .3561464E-04 .1891901E-06 -.4263255E-13

5.000000 .1071989E-03 .2956720E-00 -.4203255E-13

APPENDIX F

REFERENCES

1. R. H. Blazek, R. E. Thomas, R. K. Thatcher and J. L. Easterday,
"Tabular System Reliability Analysis", Battelle, Columbus Lab.
AFFDL-TR-71-128.
2. "An Automatic Reliability Mathematical Model", Boeing Document No. D6A-10500-1.
3. Y. W. Ng and A. Avizienis,
"A Unifying Reliability Model for Closed Fault Tolerant Systems"
1975 Int. Symp. on Fault Tolerance Computers, Paris June 18-20.
4. Jean-Claude Laprie
"Reliability and Availability of Repairable Structures"
1975 Int. Symp. on Fault Tolerance Computers, Paris, June 18-20.

SOLUTION OF A DIAGONAL MARKOV EQUATION SYSTEM

The system to be solved is:

$$\dot{\bar{p}} = \Lambda^T \bar{p} \quad \text{with } \bar{p}(0) \text{ known}$$

$$\text{and } \Lambda = \begin{bmatrix} \lambda_{11} & \lambda_{12} & \lambda_{13} & \dots & \lambda_{1n} \\ 0 & \lambda_{22} & \lambda_{23} & \dots & \lambda_{2n} \\ 0 & 0 & & \dots & \\ & & & & \lambda_{nn} \end{bmatrix}$$

Find a matrix, T, such that:

$$T^{-1} \Lambda^T T = D$$

$$\text{where } D = \begin{bmatrix} \lambda_{11} & & & 0 \\ & \lambda_{22} & & \\ 0 & & \ddots & \\ & & & \lambda_{nn} \end{bmatrix} \quad (\text{Diagonal})$$

Then we can make the transformation

$$\bar{q} = T^{-1} \cdot \bar{p}$$

$$\therefore \dot{\bar{q}} = T^{-1} \Lambda^T T \bar{q} = D \bar{q}$$

$$\text{with the solution } q_i(t) = e^{-\lambda_{ii} t} \cdot q_i(0)$$

$$\text{or } \bar{q}(t) = \text{EXP} \cdot \bar{q}(0)$$

where $\text{EXP} = \text{DIAGONAL WITH ELEMENTS} = e^{-\lambda_{ii} \cdot t}$

$$\therefore \underline{p(t) = T^{-1} q(t) = T^{-1} \cdot \text{EXP} \cdot T p(0)}$$

T may be determined from:

$$\Lambda^T \cdot T = T \cdot D$$

$$\begin{bmatrix} \lambda_{11} & 0 & & 0 \\ & \lambda_{22} & & \\ \lambda_{12} & & \ddots & \\ \lambda_{13} & & & \lambda_{n-1, n-1} \\ \lambda_{1n} & \lambda_{2n} & & \lambda_{nn} \end{bmatrix} \times \begin{bmatrix} t_{11} & & & 0 \\ & t_{22} & & \\ t_{21} & & \ddots & \\ t_{31} & & & \\ t_{n1} & \dots & & t_{nn} \end{bmatrix} =$$

$$= \begin{bmatrix} t_{11} & & & 0 \\ t_{21} & & \ddots & \\ \vdots & & & \\ t_{n1} & \dots & & t_{nn} \end{bmatrix} \times \begin{bmatrix} \lambda_{11} & & & \\ & \lambda_{22} & & \\ & & \ddots & \\ 0 & & & \lambda_{nn} \end{bmatrix}$$

Setting elements i, j equal we get:

$$\sum_{k=1}^n \lambda_{ki} \cdot t_{kj} = \lambda_{jj} t_{ij}$$

and since $\lambda_{ki} = 0, k > i$ and $t_{kj} = 0, k < j$:

$$\sum_{k=j}^i \lambda_{ki} t_{kj} = \lambda_{jj} t_{ij}$$

$$\therefore \sum_{k=j}^{i-1} \lambda_{ki} t_{kj} + \lambda_{ii} t_{ij} = \lambda_{jj} t_{ij}$$

$$t_{ij} = \frac{1}{\lambda_{jj} - \lambda_{ii}} \sum_{k=j}^{i-1} \lambda_{ki} \cdot t_{kj}$$

t_{ij} , $i > j$ may be solved by recursion, arbitrarily starting with $t_{jj} = 1$, i.e.

$$t_{j+1, j} = \frac{1}{(\lambda_{jj} - \lambda_{j+1, j+1})} \lambda_{j, j+1} \cdot t_{jj} \quad \text{Etc.}$$

APPENDIX G

ARCS COVERAGE STUDY REPORT

(General Electric Project Memorandum 76-AR-01)

PURPOSE

The purpose of this memo is to provide a final report on GE's coverage study activities as required by SOW paragraphs 3.2-4 and 3.6-5.

INTRODUCTION

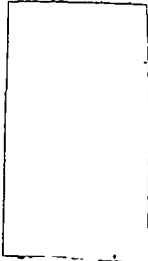
Using the standard terminology adopted for the ARCS study (reference 1), coverage is generally defined as "the conditional probability that a stage continues to perform the required function(s) given a failure (permanent fault)." Note that transient faults are excluded from this definition, and that the terms "failure" and "permanent fault" are equivalent within the ARCS vocabulary. The specific coverage definitions of interest for ARCS have been obtained by applying the above conceptual definition to each of the stages within a detailed ARCS reliability model. In this context, there is an array of coverage values, or parameters, which is of critical importance in the determination of the ARCS functional availability and mission reliability. For this reason, a significant portion of GE's ARCS effort has been concerned with the problem of evaluating fault coverage. The task outline shown in figure G-1 indicates the scope of the coverage study activity.

Most of the significant results of the coverage study have been presented previously at various ARCS oral reviews with the aid of viewgraphs. The discussion which follows is built upon much of the earlier viewgraph material and is expanded in some areas. Generally speaking, the discussion follows the task outline in figure G-1.

- 13438
1. Review the ARCS analytical model definitions for coverage and establish precise hardware-oriented interpretations.
 2. Identify and review the state-of-the-art in terms of electronic component failure rate prediction, component failure modes, and the failure complexities of MSI and LSI digital microcircuits.
 3. Establish the rationale and develop the mathematical theory which supports the use of statistical estimation as a practical coverage assessment technique.
 4. Select one ARCS triplex stage which is of suitable design complexity, and,
 - a. build a simple laboratory test breadboard of the stage
 - b. evaluate stage coverage parameters using both paper analysis and statistical estimation based upon manual fault insertion.
 5. Evaluate the breadboard coverage study results and formulate final conclusions. Prepare a final report in engineering memo form and submit to Boeing.

GE's Coverage Study Plan - TASK OUTLINE

Figure G-1



DISCUSSION

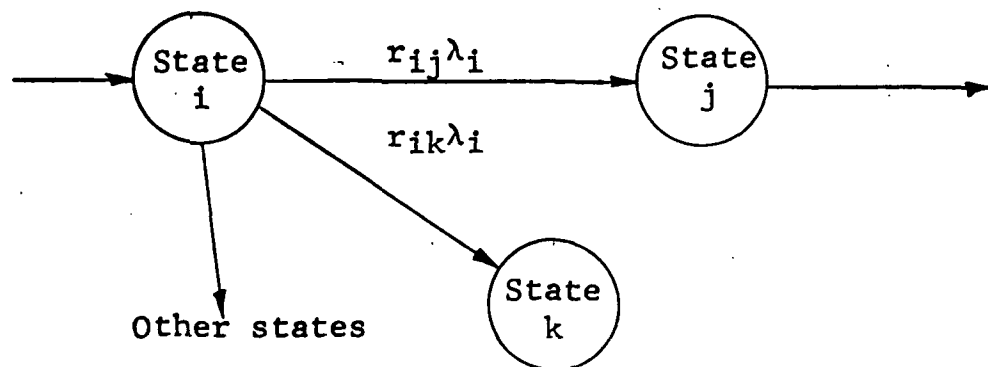
The ARCS reliability model is based upon a discrete-state, continuous-time, Markov model for each triplex stage (3 identical modules operating in parallel) within the system. With this modeling approach it is possible to define fault coverages in terms of Markov states and transitions. Figure G-2 provides an illustration of the basic concepts involved. For each Markov state i there is a corresponding fault rate λ_i which determines the probability of exiting the state due to the occurrence of a (single) fault. Specifically, the amount of time the stage spends in state i is an exponentially distributed random variable with mean equal to the reciprocal of λ_i . In general, the fault rate λ_i includes both permanent and transient faults. However, in this discussion of coverage the term λ_i will be restricted to include only the permanent fault rate. (Transient faults are covered by a composite Markov model which uses "leakage" parameters in a similar fashion to coverage parameters).

Upon exiting state i the stage may transition to any number of other Markov states. The fault rate associated with each possible transition path is a fixed proportion of the total exit rate from state i . Thus, the failure rate ratio term, r_{ij} , defined in figure G-2 is a direct measure of the proportion of λ_i which is assigned to the i to j transition.

For a given stage there is a collection of particular states, the set S , for which the stage successfully performs its required function(s). Relative to any state i contained in S , the coverage C_i is defined as,

$$C_i = P \left\{ \begin{array}{l} \text{Continued stage success} \\ \text{Transition from state } i \\ \text{due to a permanent fault} \end{array} \right\}$$

which may be expressed as a sum of failure rate ratios as follows,



Note:

All transition arrows represent transitions due to permanent faults only.

λ_i = total permanent fault exit rate from state i

Failure Rate Ratio $r_{ij} = P \left\{ \begin{array}{l} \text{Transition to state } j \\ \text{Transition from state } i \\ \text{due to a permanent fault} \end{array} \right\}$

that is, $r_{ij} = \frac{\text{Permanent fault transition rate from state } i \text{ to } j}{\text{Total permanent fault exit rate from state } i}$

or,

$$r_{ij} = \frac{\lambda_{ij}}{\lambda_i}$$

Markov Model Concepts
Necessary for Coverage Definition

Figure G-2

$$C_i = \sum_{j \in S} r_{ij} \quad (1)$$

Thus, coverage relative to state i is simply a measurement of the total proportion of the exit rate λ_i which is assigned to state transitions that maintain stage success (i.e. all i to j transitions where j is contained in S). From figure G-2 it follows that C_i may be expressed alternately as,

$$C_i = \sum_{j \in S} \frac{\lambda_{ij}}{\lambda_i} \quad (2)$$

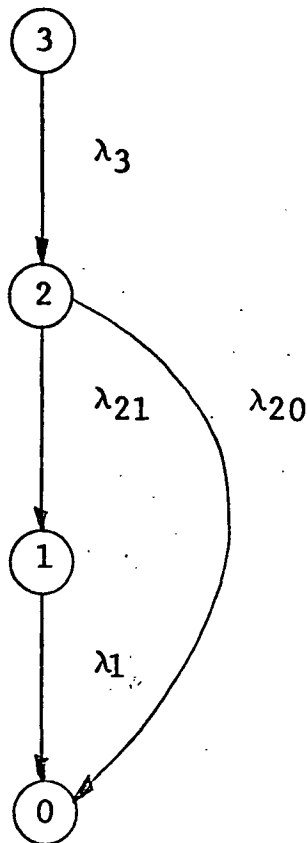
For most of us, these equations do not provide much intuitive feeling as to the real hardware oriented meaning of coverage. Perhaps the first obstacle is the definition of the Markov states relative to a particular triplex set of modules. For discussion, let's consider the four most obvious states for a triplex module set: operating with none failed (triplex), operating with one failed (duplex), operating with two failed (simplex), and not operating and/or three failed (stage failure). If the failure rate of each module is λ , then the exit rate from the triplex state is 3λ . Similarly, the exit rate from the duplex state is 2λ , and from the simplex state is λ . Now, suppose that some proportion, r_{21} , of the exit rate from the duplex state corresponds to a transition to simplex operation, and the remaining proportion, $1 - r_{21}$, corresponds to stage failure. Figure G-3 provides a state transition diagram for this example.

Triplex
operating
state

Duplex
operating
state

Simplex
operating
state

Stage
Failure



λ = module failure rate

$$\lambda_3 = 3\lambda$$

$$\lambda_2 = 2\lambda$$

$$\lambda_{21} = 2\lambda r_{21}$$

$$\lambda_{20} = 2\lambda(1-r_{21})$$

$$\lambda_1 = \lambda$$

$$C_1 = \frac{\lambda_3}{\lambda_3} = 1$$

$$C_2 = \frac{\lambda_{21}}{\lambda_2} = r_{21}$$

Simple Triplex Stage Model

Figure G-3

It follows immediately from equation (2) and figure G-3 that,

First failure coverage $C_1 = 1$

Second failure coverage $C_2 = r_{21}$

This simple triplex example has so far shown that a Markov stage model may be used in a rather straight-forward manner to obtain definitions for two important coverage parameters. However, these definitions are still somewhat lacking in intuitive appeal. The most common difficulty in this area is to relate the more familiar concepts of percent fault detection and isolation to the concept of coverage. In the preceding example this involves just the duplex operating state, where an additional failure can only be tolerated if it is detected and isolated, so that successful redundancy degradation to simplex operation is achieved. Through the use of comparison monitoring it is easy to see how each module in the duplex state can detect a fault occurrence. Further, with the addition of a self-monitoring capability within each module, it is evident that some percentage of faults local to a module will be successfully isolated. Thus, in this example at least, the intuitive point-of-view might be that second failure coverage, and the percentage of second failures that are successfully detected and isolated, may be somewhat the same thing. It turns out that there is a very simple relationship between these two measures of fault tolerance which is of critical importance in the fault coverage evaluation problem. This relationship can be developed with the aid of the following definitions; let,

$\bar{\lambda}_{ij}$ = the mean failure rate of the N permanent faults which can cause a transition from state i to j

$\bar{\lambda}_i$ = the mean failure rate of the total number M of permanent faults which can cause an exit from state i .

These definitions of course assume that the number of possible permanent faults in the modules under consideration is finite. For a given state i it follows from figure (2) that r_{ij} may be expressed in terms of the above mean failure rates as,

$$r_{ij} = \left(\frac{N}{M} \right) \cdot \left(\frac{\bar{\lambda}_{ij}}{\bar{\lambda}_i} \right) \quad (3)$$

The ratio (N/M) is a positive fraction (a number between 0 and 1) since N is always less than or equal to M . The ratio $(\bar{\lambda}_{ij}/\bar{\lambda}_i)$ is a positive number which, in general, may be greater or less than one. The product of the two ratios is guaranteed to be a number less than or equal to one, because $N \bar{\lambda}_{ij} = \lambda_{ij}$, and this term is always less than or equal to $M \bar{\lambda}_i = \lambda_i$. For discussion purposes it is convenient here to consider the simple case where in equation (2) $C_i = r_{ij}$. Then, the ratio (N/M) expressed as a percentage may be viewed as the percent of "covered" failures, meaning those failures that are detected and isolated. So that one may write,

$$\%r_{ij} = (\% \text{ of "covered" failures}) \cdot \left(\frac{\bar{\lambda}_{ij}}{\bar{\lambda}_i} \right)$$

Expressed this way, it is clear that the ratio of mean failure rates provides a weighting factor which increases or decreases the actual $\% r_{ij}$. Thus, the most common intuitive view of coverage may be placed in agreement with a Markov modeling approach by the simple addition of a failure rate weighting factor.

At this point, the central issues of the coverage evaluation problem are in evidence. To evaluate any C_i parameter it is necessary to know the following:

- the total number (M) of possible permanent faults in the modules under consideration.
- the failure rate corresponding to each permanent fault.
- the transition result from each permanent fault occurrence.

[]

Knowing these things the terms N , $\bar{\lambda}_{ij}$ and $\bar{\lambda}_i$ are known for each r_{ij} , and hence for C_i . The balance of this memo describes GE's study efforts to find practical ways to obtain the above data.

Failure Rate Data: Sources and State-of-the-Art

There are several accepted sources from which to obtain component failure rate data. Given a complete component parts list for the module under consideration it is a simple procedure to estimate the total module failure rate. However, this data alone is not sufficient for undertaking a coverage evaluation. What is required is data concerning the failure rate attributable to each of the possible failure modes of each component. Only with this type of data can one begin to determine the total number of possible permanent faults in a module, and the failure rate associated with each fault.

An industry review was conducted to determine the availability of the needed data. The following were the primary sources considered:

- MIL-HDBK-217B (and related MIL-Specs)
- Failure Rate Data Exchange (Formerly FARADA)
GIDEP - Government-Industry Data Exchange Program
- Reliability Branch - Reliability Analysis Center (RBRAC)
USAF - Rome Air Development Center
- ARINC - Aeronautical Radio, Inc.

In the general area of military standards and specifications, figure G-4 summarizes what is available. MIL-HDBK-217B is the primary source today for electronic component failure rate data. This handbook does not provide data for component failure modes. None of the military reliability and/or component standards appear to be concerned with this type of data.

453 446

Standards Related to
the Prediction and/or
Demonstration of Equipment
Reliability

MIL-STD-785A
Reliability Program for
Systems and Equipment
Development and Production

MIL-STD-756A
Military Standard
Reliability Prediction

MIL-HDBK-217B
Reliability Prediction
of Electronic Equipment

MIL-STD-781A
Reliability Tests:
Exponential Distribution

Standards Related to
Electronic Component
Reliability (as a
primary subject)

MIL-STD-690B
Failure Rate Sampling
Plans and Procedures

MIL-STD-790C
Reliability Assurance
Program for Electronic
Parts Specifications

Established
Reliability (ER)
Specifications

Resistors:

MIL-R-39005
-39007
-39008
-39009
-39017
-55182

Capacitors:

MIL-C-14157
-19978
-39001
-39003
-39006
-39014
-39022

Figure G-4

[REDACTED]

For later reference in this memo, it is significant to note here that the "Established Reliability" (ER) specifications provide failure rate values that are guaranteed with either a standard 60% or 90% confidence level. This is the best failure rate guarantee available. MIL-HDBK-217B provides only a "point estimate" with unknown confidence for component failure rates.

The FARADA/GIDEP data exchange is a significant source for actual field data on component and subassembly failure rates. Unfortunately, failure mode data is not available from this source. ARINC was contacted because of their apparent activity in the general reliability area (reference 3), and their extensive involvement with commercial aircraft electronic equipment standards. While they were once (early 1960's) engaged in reliability work, they are now no longer active in this area. ARINC is therefore not a source for any failure rate data.

The Reliability Analysis Center (staffed by subcontract personnel from IIT Research Institute, Chicago, Illinois) at RADC provided the best failure mode data (reference 2) that was found in the review. This data was for microcircuit devices. Figure [G-5] shows a table which is excerpted from reference 2. As indicated in the figure, the functional failure modes for digital TTL devices may be roughly categorized as 75% "stuck-at" faults and 25% "logical" faults. This means that I/O signal pin faults corresponding to "stuck-at-1" and "stuck-at-0" conditions are the most probable TTL failure modes. The types of faults in the "logical" fault category are those for which the device performs the incorrect logic function, without any "stuck-at" condition appearing on the I/O signal pins. For example, a failure internal to the device may cause a JK flip-flop output to toggle when it should stay set to one state.

While the Reliability Analysis Center (RAC) data is in the right direction, it is hardly differentiating enough to distinguish the functional failure modes of a 4-stage synchronous binary counter from those, say, of a look-ahead carry generator. Considering the complexities of many MSI and LSI digital microcircuit devices, it is obvious that such a vague characterization as the "No Output" shown in figure [G-4] is a very gross functional failure mode description.

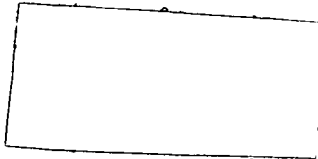
MDMR-0275 Report
TABLE I-3

BASIC TECHNOLOGY : TRANSISTOR TRANSISTOR LOGIC

<u>FAILURE INDICATORS</u>	MALFUNCTIONS		PACKAGE TYPE						
	# DEVICES	%	CAN	FPCm	FPG1	C DIP	CM DIP	E DIP	S DIP
OPEN	18	10.0					3	10	5
SHORT	32	17.8					4	19	9
INTERMITTENT	2	1.1						2	
<u>OPERATIONAL DEGRADATION</u>	128	71.1							
INTERNAL FAILURE	32	17.8						20	12
NO OUTPUT	82	45.5		2		2		34	44
FAILED TO OPERATE	1	0.6	1						
PARAMETER OUT OF TOLERANCE	13	7.2		1			9		3
TOTAL:	180								

I/O signal pin "Stuck-at" faults account for 73.3% of total. The remaining 26.7% of the faults fall into the general category of "logical faults".

Figure G-5



Nevertheless, the RAC data does provide a basis for making some assumptions for the purposes of analysis. The percentage of "stuck-at" faults may be assumed to be uniformly distributed across the I/O signal pins of a device. The percentage of "logical" faults may be treated by making subjective assessments as to the functional failure modes which are in some way "most critical" to the analysis.

The outcome of the failure rate data review may be briefly summarized with the following observations and conclusions:

- Current reliability analysis and testing of components is almost exclusively concerned with measuring and/or predicting the mean life parameter of the exponential model.
- The apparent state-of-the-art does not include the meaningful evaluation of the relative distribution of component failure modes (this is especially the case for microcircuit devices).
- The best available data for microcircuit devices is primarily oriented to physical rather than functional failure modes.
- With today's data base, the exact evaluation of coverage in terms of the failure rate of each permanent fault is impossible.


This last conclusion is significant and unavoidable; the necessary failure rate data for all types of components and their failure modes is simply not available. Further, it does not even seem reasonable to expect that such data will ever become available. It is highly unlikely that anyone will ever attempt to identify, and estimate failure rates for, all possible failure modes of many types of complex electronic devices. Clearly, to evaluate fault coverage it is going to be necessary to find ways to overcome the absence of very basic data. This problem will be addressed in a later section of this memo.

Failure Mode and Effect Analysis

In addition to the requirement to know the failure rate assigned to each permanent fault, it is necessary to know the effect of each fault. Generally the procedure known as Failure Mode and Effect Analysis (FMEA) is used to obtain such information. This procedure has a history of application to many types of systems where the performance of the system under failure conditions is important or even critical for safety reasons. Typically, FMEA involves the tabulation of all possible faults, or failure modes, and the systematic evaluation of the effects of each fault considered one at a time or, in some cases, in combinations. While the method is conceptually simple enough, in actual practice there are many difficulties which arise, especially for a complex digital system such as ARCS.

The major difficulty concerns the sheer magnitude of the task. A simple numerical example will quickly illustrate the point. Digital microcircuits are most commonly packaged in 14 or 16 leaded dual-in-packages (DIPS). Considering I/O signal pin "stuck-at" faults typically results in 24 to 28 fault possibilities per DIP, excluding the power and ground pins. When "logical" faults are added, this number may range from 24 up to 36 possible faults for the typical 14 or 16 pin DIP. A digital computer with CPU, I/O section, and memory may easily contain 500 DIPS. At an average of 30 faults per DIP the total number of faults evident on pins alone is 15,000! When boards, connectors, discrete components, power supplies, power runs and connections, and other miscellaneous parts are included, the number of possible faults in a digital computer is in the vicinity of 15,000 to 20,000 faults.

Not only is the number of faults potentially overwhelming, but in a programmable digital machine the number of possible effects ranges from the obvious to the extremely subtle. Certain faults in the hardware will have several different effects determined by the resident software, and the operating state when the fault occurs. For these reasons FMEA, as a manual paper and pencil method, is



an enormous undertaking when applied to a digital system. Computer-aided analysis and simulation is useful in some areas for FMEA, but, because of the need to consider operating software together with gate level hardware models, computer aids are quickly limited by computer execution time. For example, to simulate one second of real time operation when the gate level model uses a 4MHz. clock, requires 4,000,000 simulated clock intervals. With any complexity at all in the hardware model (say 5000 equivalent gates) it is likely that each simulated clock interval may require on the order of 25 msec of simulation running time. For 4,000,000 clock intervals this amounts to 100,000 seconds, or more than one day! Obviously, if 15,000 to 20,000 faults are considered one at a time with such a simulation, the computer running time involved is absurd.

Clearly, the use of a conventional FMEA approach for the purposes of coverage evaluation is - at least for a digital system - impractical. Consequently, the problem of coverage evaluation begins to have a rather formidable aspect to it. The required failure rate data is unavailable, and even if the data were available, the FMEA task appears overwhelming. Faced with this state of affairs, a new approach to the problem is necessary. The following sections describe GE's efforts in this direction.

Key Assumptions Leading to a Statistical Estimation Approach

Recalling equation (3) and the definitions of the associated terms, the coverage evaluation problem involves a set of M faults, with some subset of N "covered" faults which are to be appropriately identified. Since the failure rate corresponding to each fault must be considered largely unknown, it is reasonable to assume that,

The set of "covered faults" constitutes a random
sample of N failure rate values from a total
population of M failure rate values.

In other words, if the failure rate of each fault is unknown when the failure detection and isolation mechanisms are designed, it is unlikely that the coverage or non-coverage of faults will be dependent upon their failure rate. Therefore, the mean failure rate ($\bar{\lambda}_{ij}$) of the subset of N faults may be viewed as a sample mean from the failure rate population of M faults (with mean $\bar{\lambda}_i$). The "Strong Law of Large Numbers" guarantees that the sample mean approaches the population mean for large N. Further, the Central Limit Theorem guarantees that the distribution of the sample mean approaches a normal distribution for large N.

The significance of the above assumption, and its theoretical implications, is that it provides a basis for considering a simplifying approximation for coverage. This approximation is motivated by the following observations concerning r_{ij} ,

$$r_{ij} \geq \frac{N}{M} \quad \text{for } \bar{\lambda}_{ij} \geq \bar{\lambda}_i \quad (4)$$

$$r_{ij} < \frac{N}{M} \quad \text{for } \bar{\lambda}_{ij} < \bar{\lambda}_i \quad (5)$$

Equation (4) provides a lower bound on each r_{ij} term which applies whenever the indicated mean failure rate relationship is satisfied. Since coverage, C_i , is the sum of selected r_{ij} terms, the lower bound in equation (4) leads directly to a lower bound for coverage. Thus, for the simple case where $C_i = r_{ij}$, this relationship holds,

$$\frac{N}{M} \leq C_i \leq 1 \quad \text{for } \bar{\lambda}_{ij} \geq \bar{\lambda}_i \quad (6)$$

Now, it is evident from (6) that a conservative approximation for coverage may be obtained by evaluating the ratio (N/M) , provided it is known that $\bar{\lambda}_{ij} \geq \bar{\lambda}_i$. This is a very significant observation because it means that coverage may be evaluated without the need for failure rate data for each fault, if only the relative magnitude of two mean failure rates is known. On an absolute basis, of course, if the data existed to determine the mean

failure rates then the failure rate of each fault would be known as well. However, on a probabilistic basis it is possible to directly evaluate the relationship between the two mean failure rates using the key assumption stated above and, in particular, the Central Limit Theorem. In general, the probability question of interest is,

$$P \left\{ r_{ij} \geq k \left(\frac{N}{M} \right) \right\} = P \left\{ \bar{\lambda}_{ij} \geq k \bar{\lambda}_i \right\} = ?$$

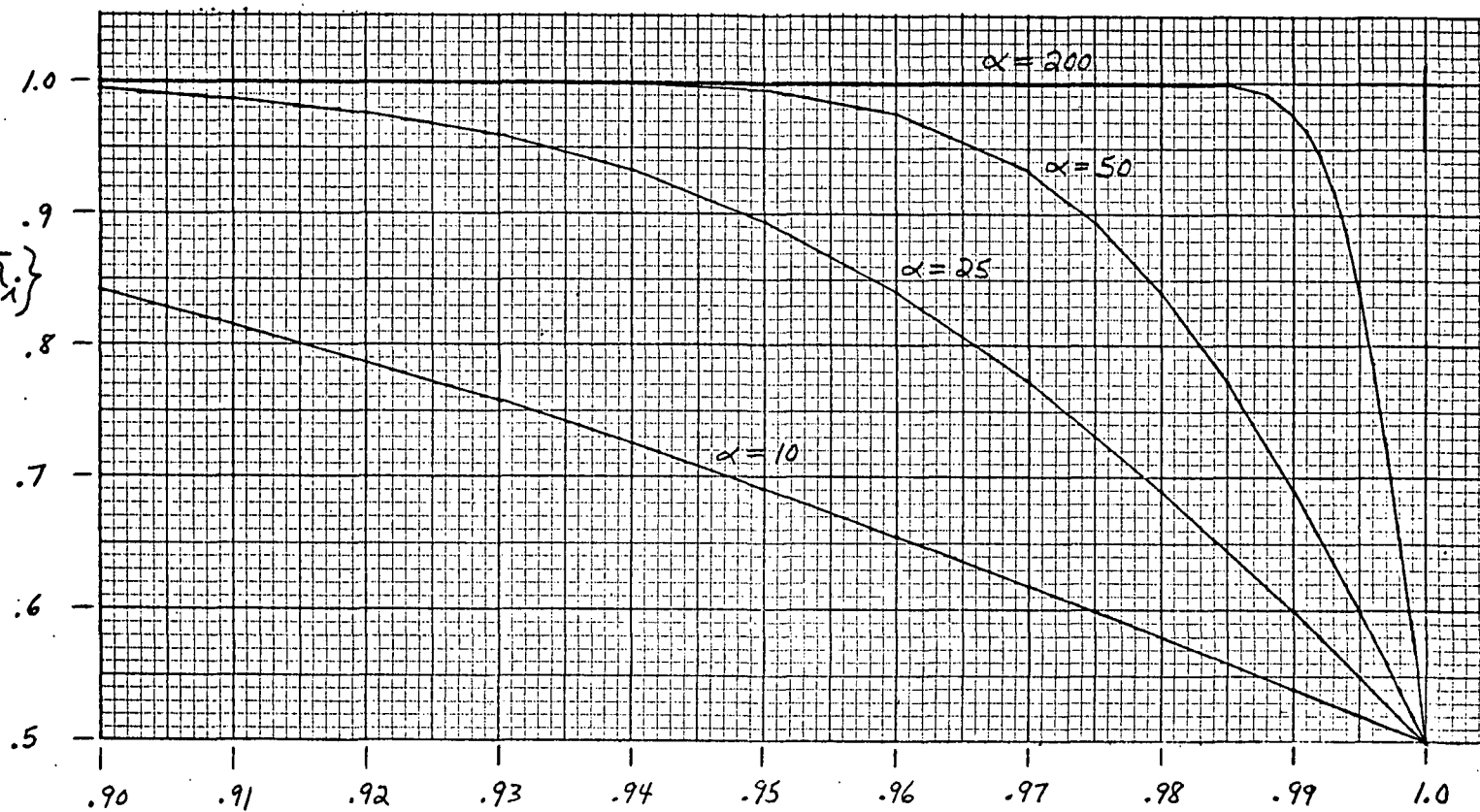
where $0 < k \leq 1$. Considering $\bar{\lambda}_{ij}$ as the mean failure rate for a sample of size N, then by the Central Limit Theorem,

$$P \left\{ \bar{\lambda}_{ij} \geq k \bar{\lambda}_i \right\} \xrightarrow{\text{large } N} P \left\{ z \geq - \frac{(1-k) \bar{\lambda}_i \sqrt{N}}{\sigma_i} \right\} \quad (7)$$

where z is the standard normal variate, and σ_i is the standard deviation of the failure rate population.

Figures [G-6] and [G-7] illustrate the numeric consequences of equation (7). In figure [G-6] the important trend to note is that as the α term increases the probability that $\bar{\lambda}_{ij} \geq k \bar{\lambda}_i$ approaches one for k values closer to one. For example, the probability that $\bar{\lambda}_{ij} \geq .99 \bar{\lambda}_i$ is .977 for $\alpha = 200$. Note, however, that regardless of the size of α all curves go through .5 probability when $k = 1$. In figure [G-7] it is apparent that as the failure rate distribution becomes more highly peaked (small variance relative to the mean) the α term stays in the order of several hundred for lower values of N. In this regard, it is critically important to recognize that the sample size N corresponds to the total number of "covered" faults (in the λ_{ij} category) for the module under evaluation. Thus, even though the number N is used in the sample context, it is not unreasonable to suggest that values of N in the order shown in figure [G-7] are possible. This is especially so when one considers that the number of faults which are typically tabulated for FMEA purposes includes many faults lumped into equivalent failure mode classifications. For example, the I/O signal pin "stuck-at"

$$P\{\bar{\lambda}_{ij} \geq k\bar{\lambda}_i\}$$



Multiplying Factor k

Figure G-6

$$\alpha = \frac{\bar{\lambda}_i \sqrt{N}}{\sigma_i}$$

Number of Faults N in λ_{ij} Category

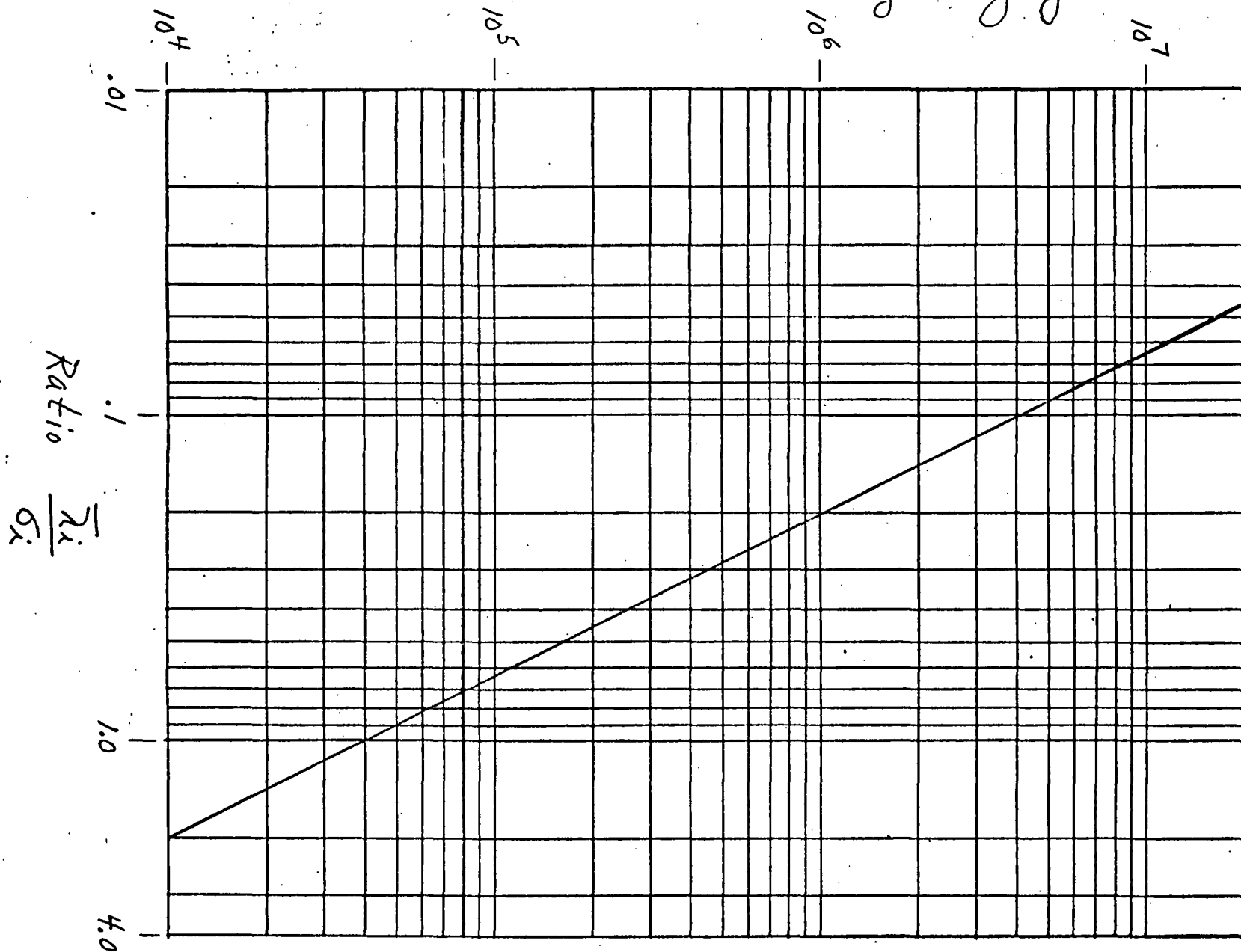
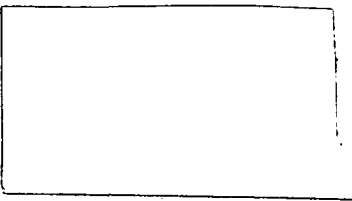


Figure G-7



faults discussed earlier are actually lumped failure modes. Any number of specific fault possibilities may be involved with the occurrence of, say, a "stuck-at-0" condition on one output pin. In this sense one may argue that the number of faults N (and necessarily M) is virtually infinite.

Certainly it is highly desirable to make the type of approximation for coverage indicated in equation (6) above. With the added multiplying factor, k , less than one, the probability is quite high that $k(N/M)$ is a conservative approximation for each term r_{ij} , and hence coverage. What constitutes a "high probability" is of course the central issue. A necessary perspective on this issue is found in the following two considerations:

- (As indicated earlier) basic component failure rate data is either given with unknown confidence, or, for established reliability parts, at a standard 60% or 90% confidence.
- Because of the large numbers of faults involved, practical analysis limitations dictate that even the ratio (N/M) would be estimated from a "sample" of faults which is assumed to constitute "all possible" faults.

In the author's opinion, the first consideration alone is sufficient to justify the use of the suggested approximation for coverage. Considering that much of the necessary failure rate data is unavailable, and that the data that is available is at best estimated to a 90% confidence, there seems to be no point whatsoever in attempting to evaluate coverage with any more than 90% confidence. The second consideration makes the additional point that even if the suggested approximation was an absolutely valid worst case approximation, the evaluation of the ratio (N/M) would, as practical matter, still take the form of an estimate.

[]

For all of these reasons the following additional assumption is accepted for the purposes of coverage evaluation:

It is assumed that in all cases $\lambda_{ij} \geq \lambda_i$ by purposeful design, or, that N is sufficiently large so that $\lambda_{ij} \geq .99 \lambda_i$ with probability approaching one.

The purposeful design aspect is always a possibility, since a designer who had any idea of the relative failure rates would try to "cover" all of the higher failure rate faults. With the two assumptions described in this section the coverage evaluation method will now be described and the results of a laboratory evaluation discussed.

The Statistical Estimation Approach

The preceeding assumptions set the stage for the evaluation of coverage in terms of the approximation that,

$$r_{ij} \geq (.99) \left(\frac{N}{M} \right) \quad (8)$$

This approximation eliminates the failure rate data problem, but still leaves the difficulty of a potentially enormous FMEA effort in order to evaluate the ratio (N/M). Clearly, in those cases where "all possible" faults, M, can be tabulated, and all of the "covered" faults, N, easily identified based upon their failure effects, FMEA is the logical and straightforward approach.

However, in the case of a digital system such as ARCS, the enormity of the FMEA task, as already described, suggests that a statistical estimation approach be employed. The essence of the approach is to use a randomly selected sample of faults to estimate the ratio (N/M) for each r_{ij} term of interest. The applicable sampling theory is derived by simply treating the ratio (N/M) as the proportion parameter of a binomial distribution. The estimation of a proportion is a standard statistical problem which has been well treated in the literature (see for example reference 4).

The necessary theory generally follows the lines that the cumulative binomial distribution can be adequately approximated by the cumulative Poisson distribution, which, in turn, can be evaluated directly in terms of a chi-squared distribution. The result is that a lower limit for the estimate of (N/M) is given by,

$$\text{Lower Limit (N/M)} = 1 - \frac{\chi^2_{1-\alpha}}{2n}$$

where,

n = fault sample size

r = number of "uncovered" faults in sample

$\chi^2_{1-\alpha}$ = Chi-squared variable with $2(r+1)$ degrees of freedom evaluated at a cumulative probability of $1-\alpha$.

Figure G-8 shows the lower limits obtained for representative sample sizes and confidence levels ($1-\alpha$).

In practice the fault sample may be selected several ways, depending mostly upon the requirements of the particular activity concerned with coverage evaluation. The sample faults may be evaluated on paper, with hardware simulation, or with fault insertion testing. The selection approach most in line with conventional FMEA is to construct a table of faults to be considered, and enumerate them. A random number generator is then used to select the desired sample. Another approach is to enumerate all components first, randomly select a sample of components, then randomly select the fault within each selected component which is to be used in the final fault sample. This approach has the advantage of not requiring the construction of a complete fault table. To keep this approach from introducing any bias in the results, it would be necessary to assume for selection purposes that the relative number of faults per component is proportional to its total failure rate; or, each component would have to be weighted for selection according to the actual number of faults which would have been used in the FMEA table approach.

Fault Sample Size n

Number of "uncovered" faults in sample	n = 100		n = 300		n = 500	
	60% Limit	90% Limit	60% Limit	90% Limit	60% Limit	90% Limit
0	.991	.977	.997	.992	.998	.995
1	.980	.961	.993	.987	.996	.992
2	.969	.947	.990	.982	.994	.989
3	.958	.933	.986	.978	.992	.987
4	.947	.920	.982	.973	.989	.984
5	.937	.907	.979	.969	.987	.981

Figure G-8 - Lower Limits for (N/M)
at 60% and 90%
Confidence Levels.

Laboratory Evaluation

In order to demonstrate the statistical estimation method and compare it to the FMEA approach, an ARCS triplex stage (which was tractable for FMEA) was selected for laboratory breadboard evaluation of coverage. The stage was the servo stage including the triplex servo actuator, servo electronics, and computer output section shown in figure G-9. A quadruplex 680J force-summed actuator with one channel bypassed was used for the triplex actuator. Analog servo electronics (per reference 1) were built up, and a (Intel 8008) microprocessor setup with real-time I/O capability was used to simulate one ARCS computer channel.

The details of breadboard operation and the coverage evaluation are presented in the form of the viewgraphs used previously to report these results. Figures G-11 through G-25 contain these viewgraphs.

As can be seen by the results in figure G-25 a second failure coverage of approximately .95 at 90% confidence was obtained -- not including the additional .99 factor in equation (8). (When this factor is added the coverage is reduced to about .94). In all cases shown in figure G-25 the statistical results are more conservative than the FMEA point estimates.

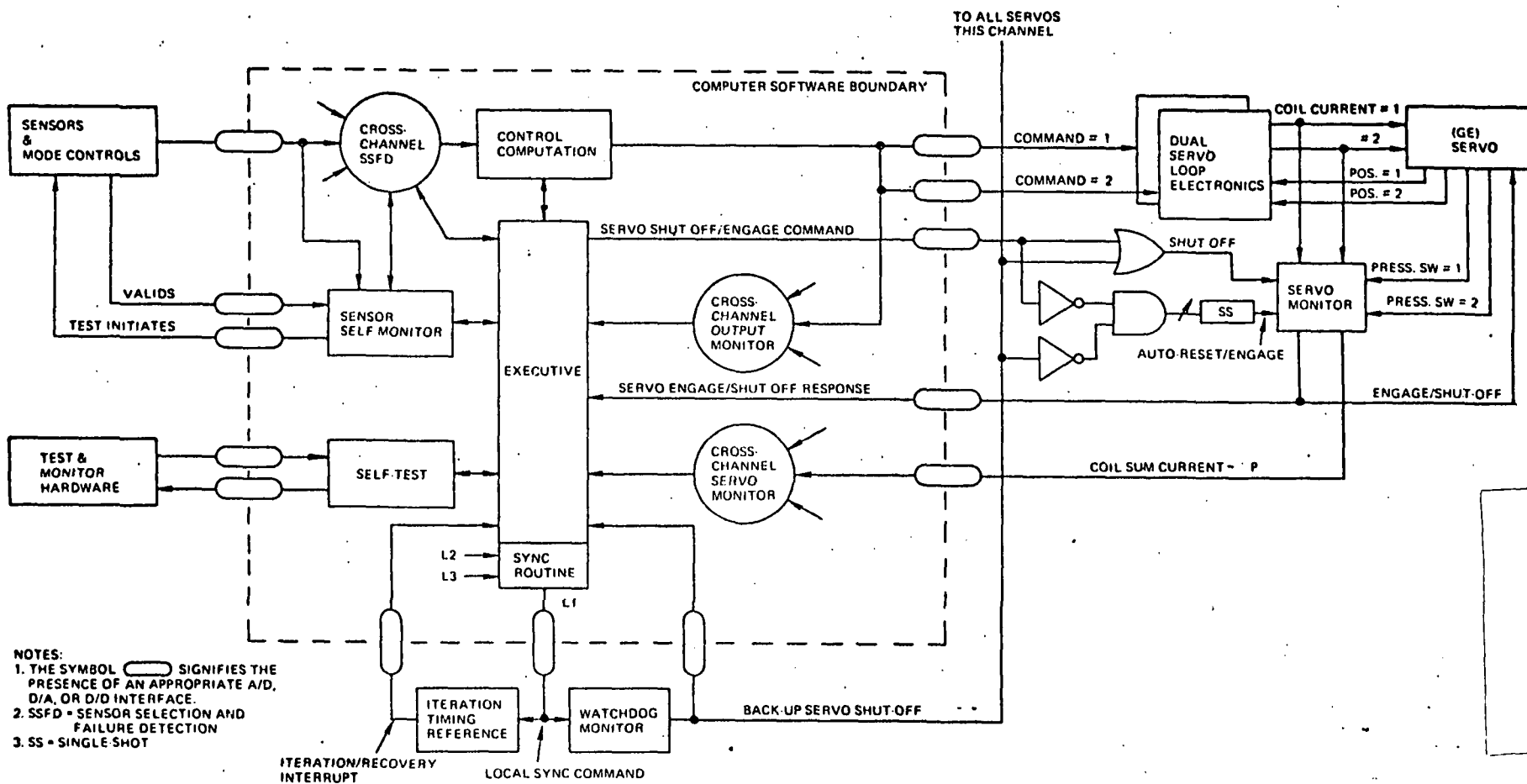
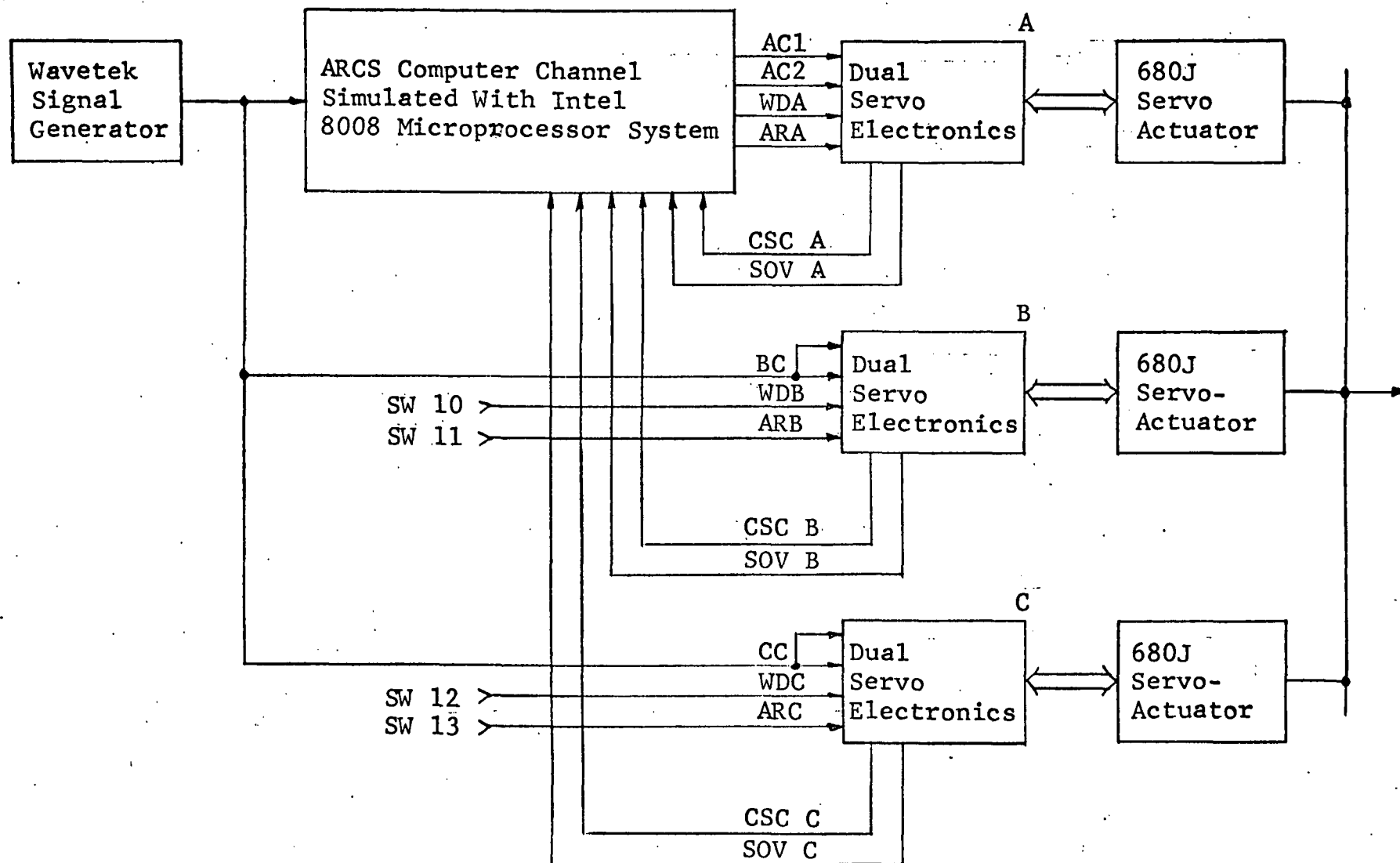


Figure G-9. ARCS Redundancy Management Block Diagram



Laboratory Breadboard Block Diagram

Figure G-11

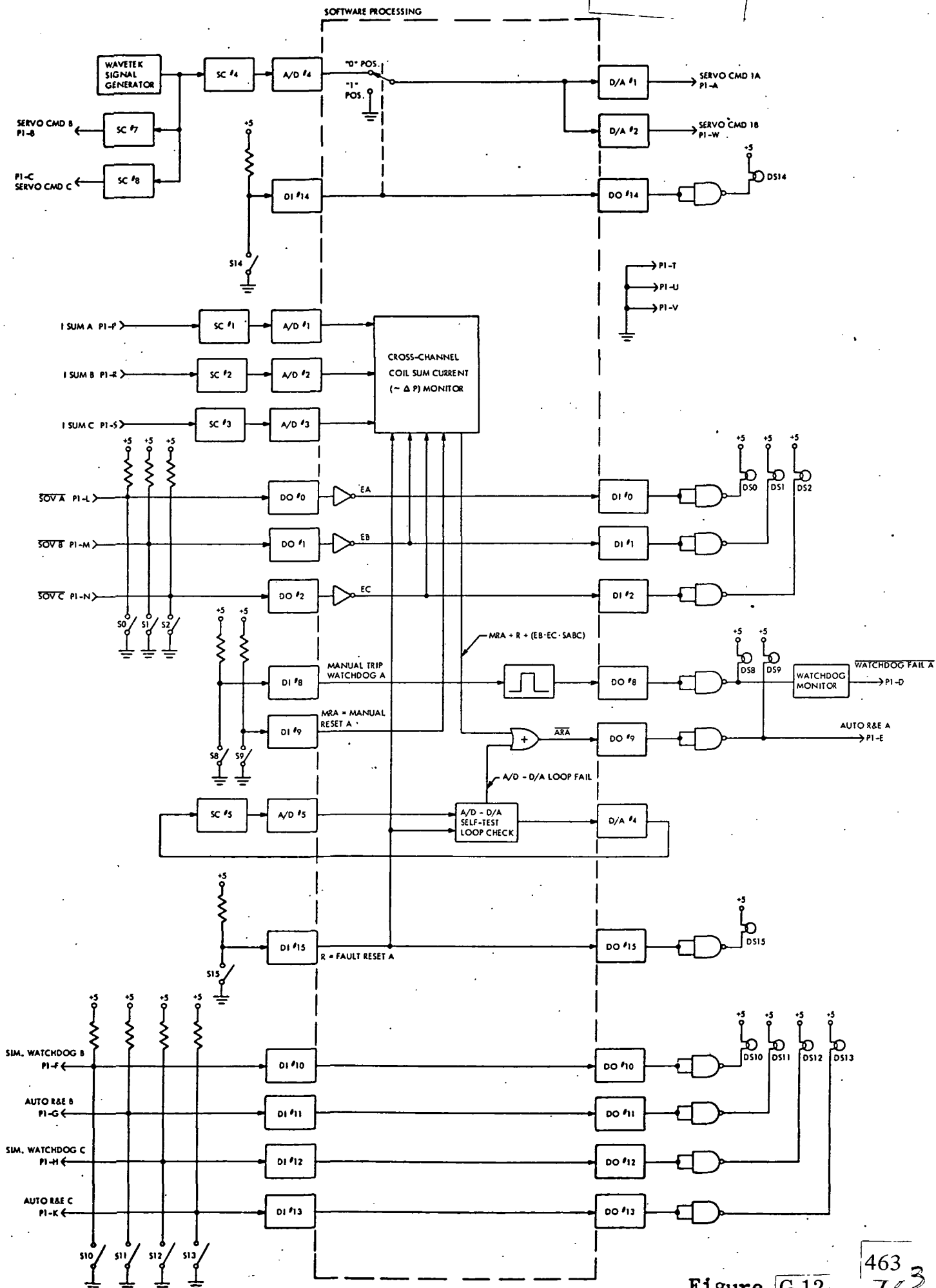
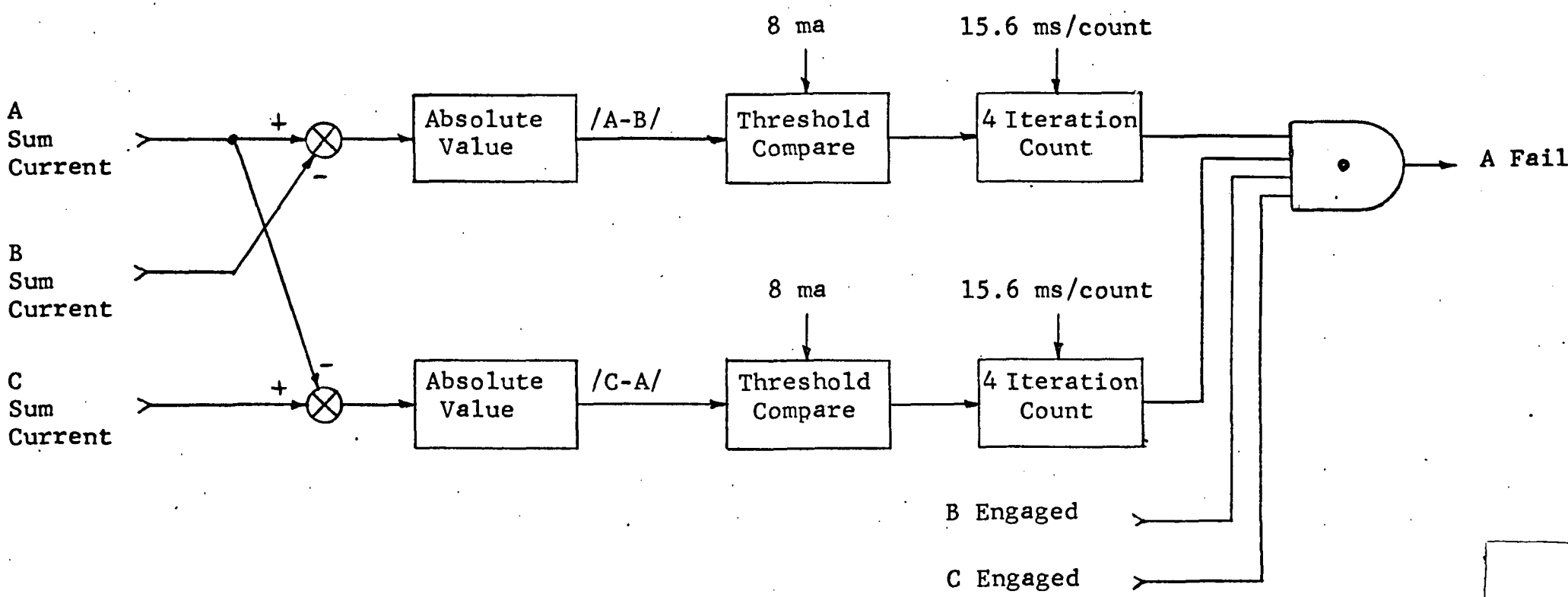


Figure G-12

464

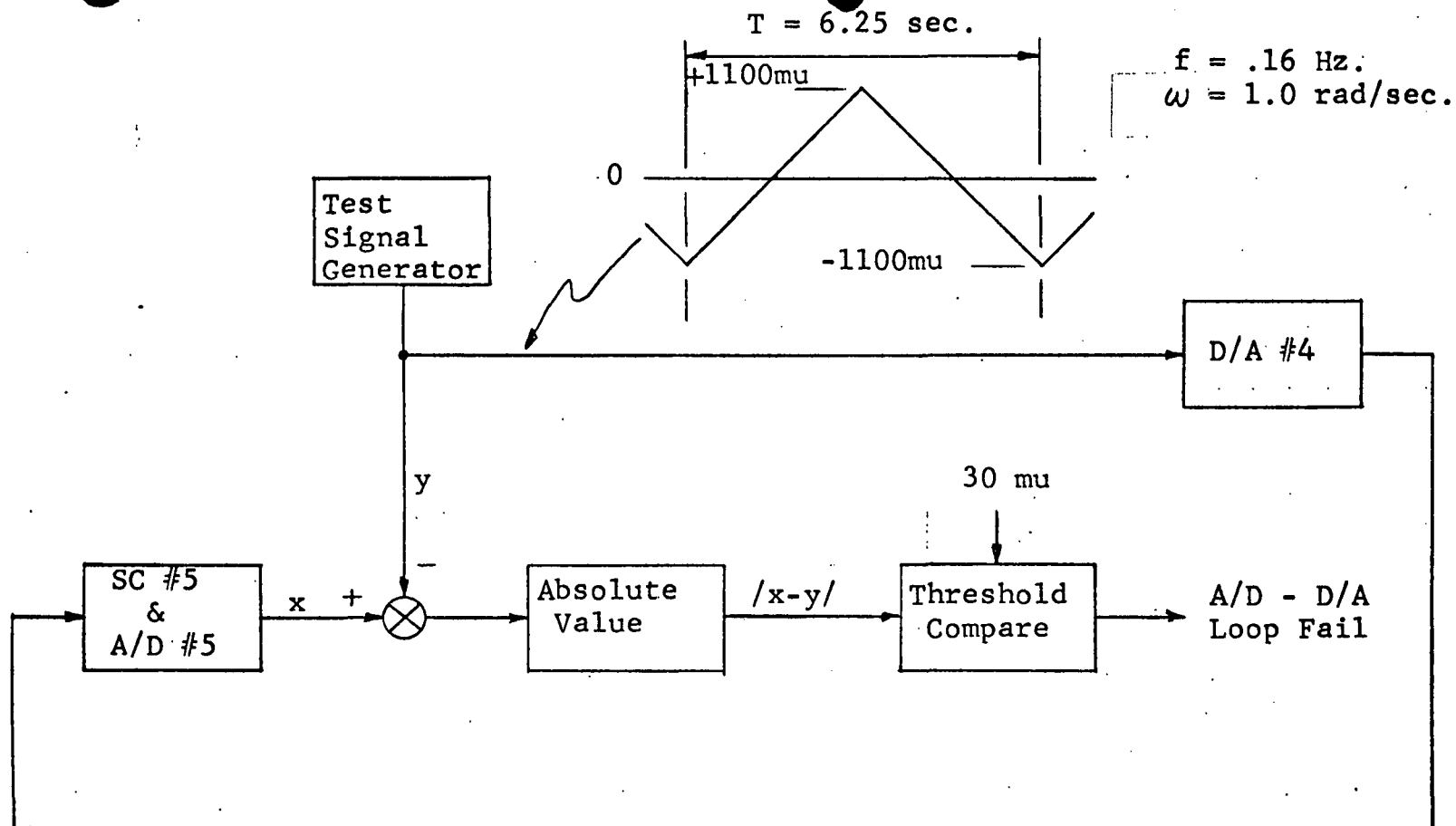


Failure Threshold = $(8 \text{ ma})(1\text{v/ma})(2048 \text{ mu/10v}) \rightarrow 1639 \text{ mu}$

Failure Threshold Referred to Servo Command Input = $\left(\frac{8 \text{ ma}}{30.2 \text{ ma/v}} \right) (2048 \text{ mu/10v}) \rightarrow 53 \text{ mu}$

Cross-Channel Coil Sum Current Monitor

Figure G-13

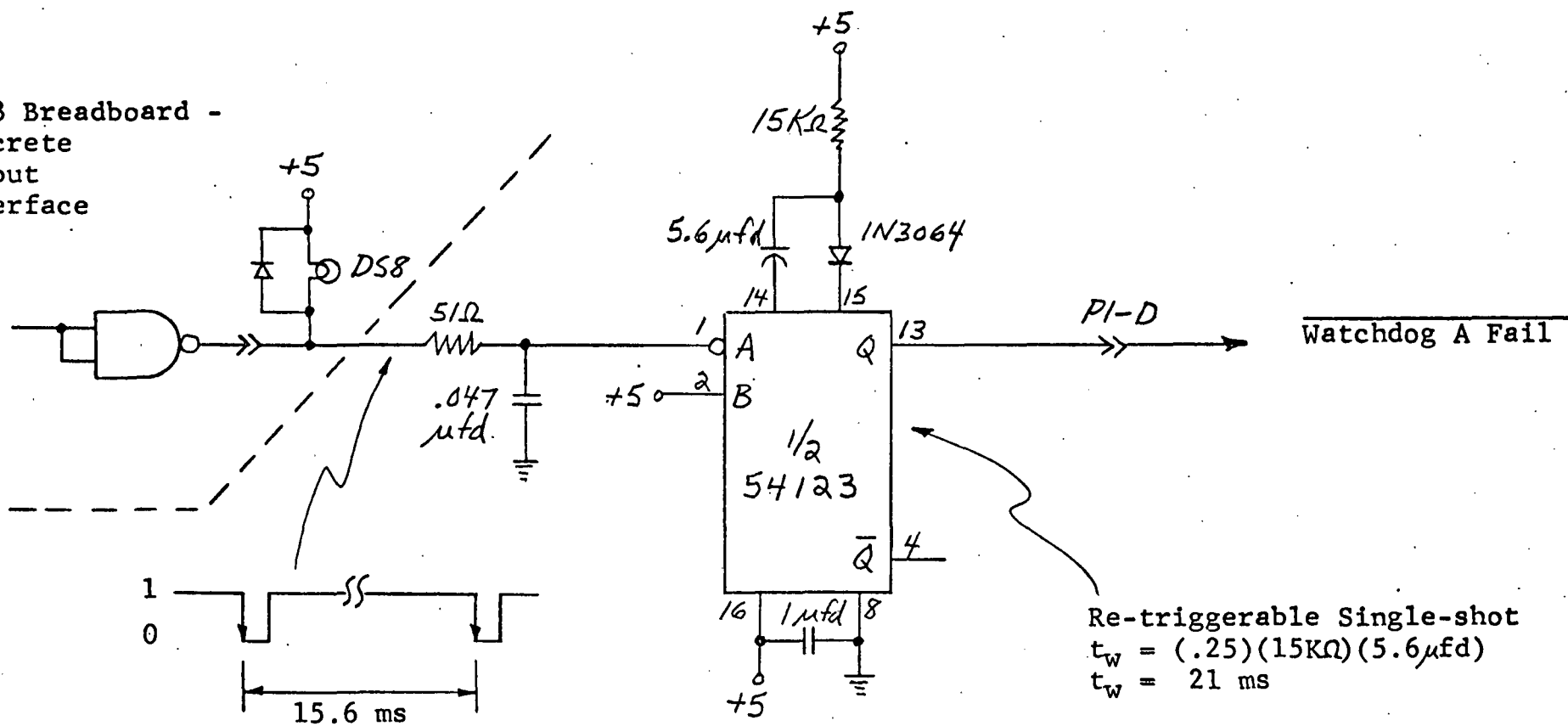


$$\text{Failure Threshold in \% of F.S.} = \frac{\pm 30 \text{ mu}}{2048 \text{ mu}} \times 100\% = 1.46\%$$

A/D - D/A Self-Test Loop

Figure G-14

8008 Breadboard - Discrete Output Interface



Watchdog Monitor

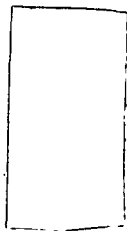
Figure G-15

468

Breadboard Demonstration and Analysis Objectives

- To verify the ARCS servo stage design concept through laboratory test.
- To evaluate second failure coverage (the duplex-to-simplpex transition rate ratio) for the ARCS servo stage using two methods:
 - a. Conventional FMEA based upon "best available" data and/or "reasonable assumptions" concerning the assignment of failure rates to each component failure mode.
 - b. Statistical estimation based upon manual insertion of randomly selected faults and the use of an approximate math model.
- To review results and draw conclusions concerning the coverage evaluation methodology.

Figure G-17



Groundrules for FMEA and Fault Table

- Only those faults within the breadboard servo stage dependency structure are considered.
- Faults are considered one at a time for a "typical" duplex operating condition.
- Failure mode descriptions are derived for the "best available" data, and in the absence of any data, are based upon "reasonable" engineering assessments.

Figure G-18

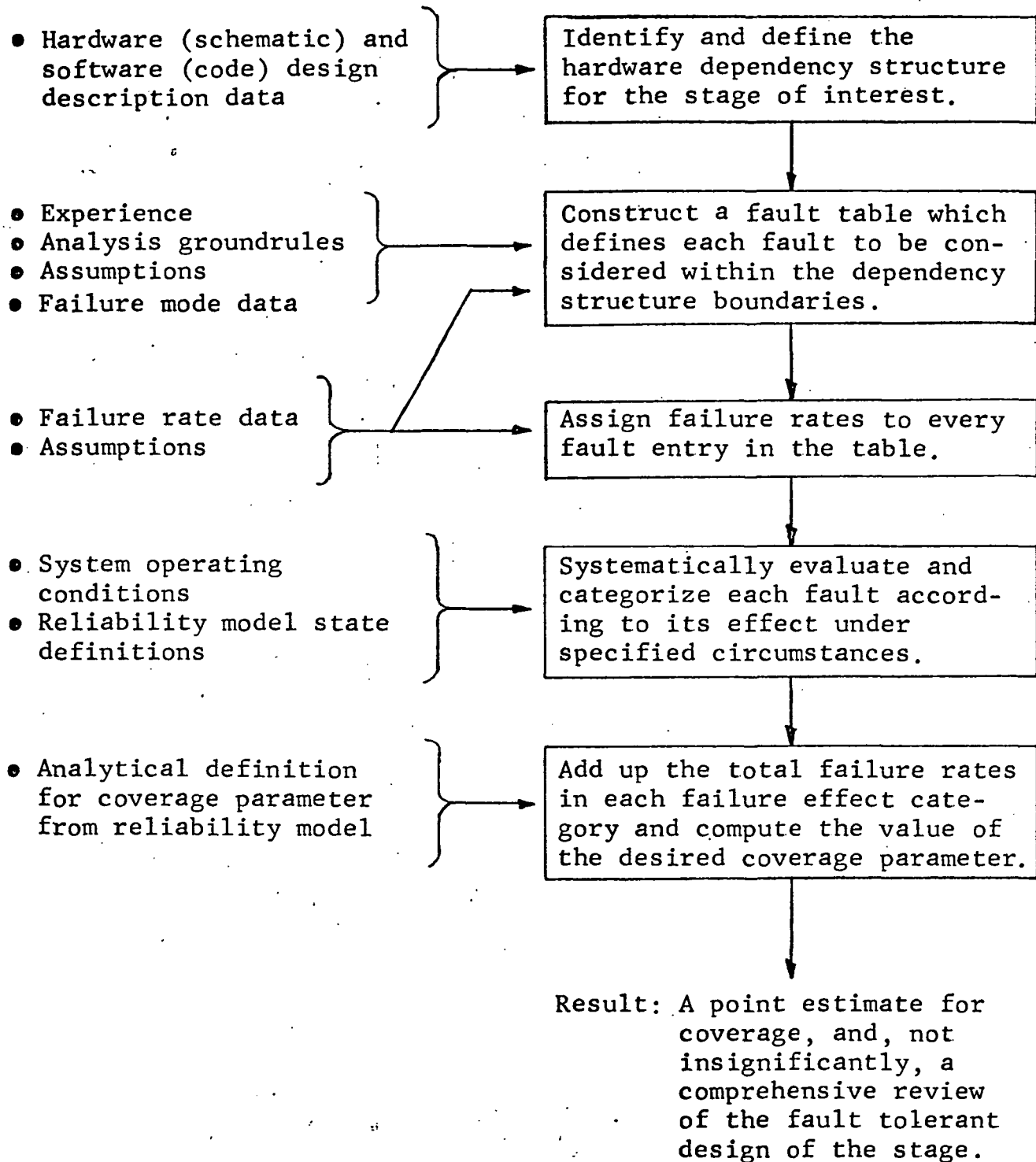
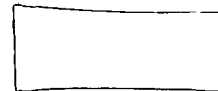


Failure Modes Considered in Fault Table

Resistors:	Open (99%) and Short (1%)
Capacitors:	Open (1%) and Short (99%)
Diodes:	Open (33%) and Short (67%)
Transistors:	Collector-to-Emitter Short (25%) Collector-to-Emitter Open (75%)
Op Amps:	Hardover Positive Output (20%) Hardover Negative Output (20%) Low gain/Open output (50%) Null Shift (10%)
TTL Logic:	I/O Pin "stuck-at" Faults (75%) Logical Fault on Output (25%)
Connectors:	Single Pin Pair Open (100%)

- Large modules and all other devices treated according to the best available data or according to subjective assessment of functional fault modes.
- Loss of ground or +5V power connections on TTL logic devices was arbitrarily excluded.

Figure G-19



Outline of the Failure Modes and Effects Analysis (FMEA) Method

Figure G-20

Figure G-21



- Hardware (schematic) and software (code) design description data

Identify and define the hardware dependency structure for the stage of interest.

- Experience
- Analysis groundrules
- Assumptions
- Failure mode data

Construct a fault table which defines each fault to be considered within the dependency structure boundaries.

- Random number generator

Enumerate the faults and select a random sample of appropriate size.

- System operating conditions
- Reliability model state definitions

Manually insert each sample fault into the hardware and categorize it according to its effect under specified circumstances.

- Analytical definition for coverage parameter from reliability model
- Statistical model for sample data

Using the sample results compute a lower limit for the desired coverage parameter at a selected confidence level.

Result: An interval estimate for coverage with a selected confidence, and a test sample evaluation of actual hardware performance.

Outline of Statistical Estimation Method

218-474

<div style="display: inline-block; transform: rotate(-45deg); text-align: center;"> Failure Effect Categories Hardware Function </div>	F ₀ Servo Failed AND Bypassed		F ₁ Servo Failed AND NOT Bypassed		F ₂ Servo NOT Failed AND Bypassed		F ₃ Servo NOT Failed AND NOT Bypassed	
	N ₀	λ ₀	N ₁	λ ₁	N ₂	λ ₂	N ₃	λ ₃
I/O Multiplexing (XA2)	271	2.364	6	.031	245	1.849	325	2.737
D/A Converter (XA6)	220	16.676	-	-	56	7.806	184	9.258
A/D Converter (XA7)	6	.025	-	-	128	9.723	63	1.483
DC Signal Conditioners (XA8)	4	.028	-	-	29	1.073	35	.357
Watchdog Monitor	1	.006	-	-	12	.119	10	.276
Servo Electronics	106	5.019	5	.069	83	1.909	96	3.482
Servo Actuator	9	37.000	4	.500	3	5.822	1	1.941
TOTALS:	617	61.118	15	.600	556	28.301	714	19.534

Total Faults = 1902

Total Failure Rate = 109.553 (failures/million hours)

$$\frac{N_0 + N_2}{N_0 + N_1 + N_2} = \frac{1173}{1188} = .987$$

$$\frac{\lambda_0 + \lambda_2}{\lambda_0 + \lambda_1 + \lambda_2} = \frac{89.419}{90.019} = \underline{\underline{.993}}$$

Results of FMEA for Duplex-to-Simplex Transition Case

Figure G-23

Hardware Function \ Failure Effect Categories	F ₀	F ₁	F ₂	F ₃	% of Faults in Sample	Actual % of Faults in Fault Table (population)
I/O Multiplexing (XA2)	22	-	14	27	40.4%	44.5%
D/A Converter (XA6)	20	-	7	14	26.3%	24.2%
A/D Converter (XA7)	2	1	8	2	8.3%	10.4%
DC Signal Conditioners (XA8)	1	-	4	4	5.8%	3.6%
Watchdog Monitor	-	-	-	-	-	1.2%
Servo Electronics & Actuator	15	1	5	9	19.2%	16.1%
TOTALS:	60	2	38	56	100.0%	100.0%
	100 faults which constitute the desired sample size					
	156 sample faults actually inserted					

Data From Random Sample Fault Insertion
for Duplex-to-Simplex Transition Case

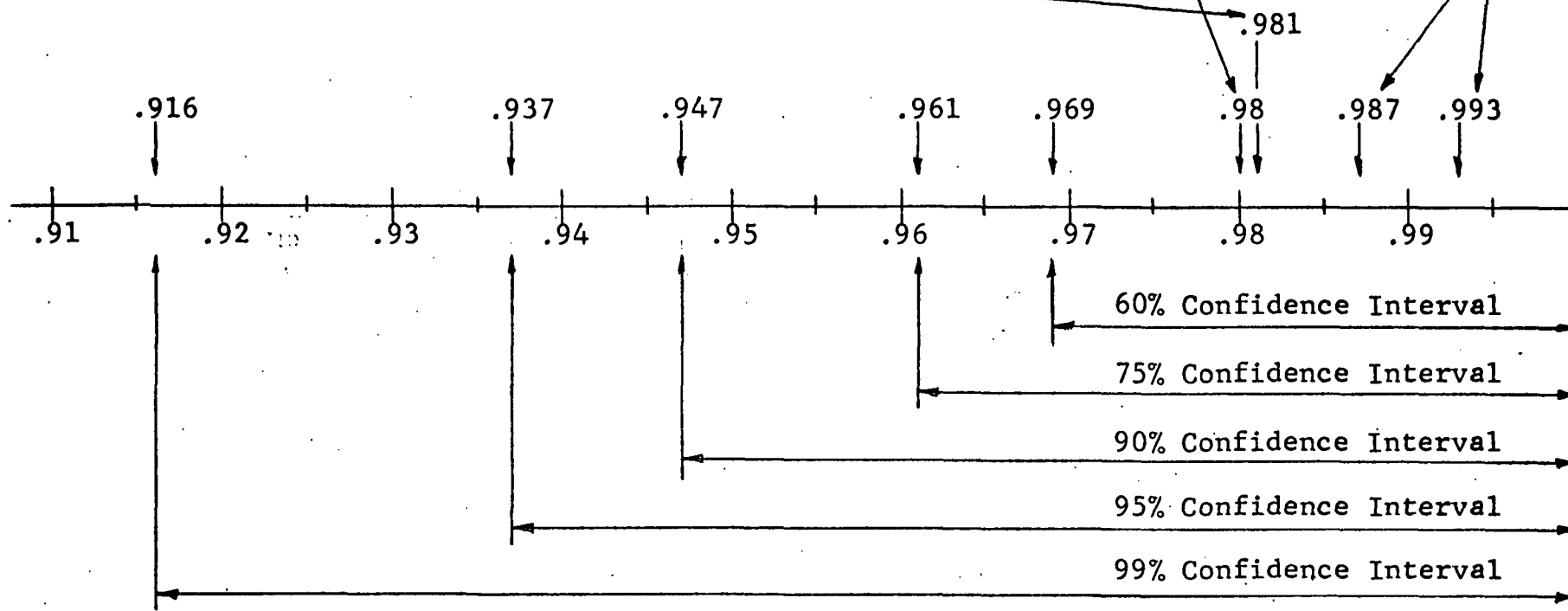
Figure G-24

48476

95% Confidence Limit Obtained by
Treating FMEA Fault Table as a
Large Random Sample Itself

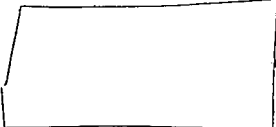
Random Sample
Point Estimate

FMEA Point
Estimates



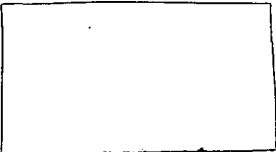
Overall Results for Servo Stage Second Failure Coverage Evaluation

Figure G-25



Conclusions

A practical method for evaluating fault coverage has been developed and demonstrated. Estimation of fault coverage at a 90% confidence level is consistent with the best failure rate data available today. Estimation of very high coverages (greater than .97) is not easily done with a high confidence, without a very large sample size. Because of the .99 multiplying factor used in the development, coverages estimated using the described method must necessarily be less than .99.



References

1. "ARCS Hardware Description", ACS 10936, General Electric Company, Binghamton, New York, December 1975.
2. "Microcircuit Device Malfunction Report", MDMR-0275, Reliability Analysis Center, RADC, Griffiss AFB, Rome, New York, March 1975.
3. "Reliability Engineering", ARINC Research Corporation, W. H. Von Alven - Editor, Prentice-Hall, Englewood Cliffs, New Jersey, 1964.
4. A. Hald, "Statistical Theory with Engineering Applications", John Wiley & Sons, Inc., New York, 1952.

APPENDIX H

RELIABILITY RESULTS DETAILS/PARAMETERS

1.0 SCOPE

This appendix gives an account of the more detailed analysis performed to establish the various Markov model parameters. Sources for these parameters are identified in Table H-1.

2.0 PERMANENT FAILURE RATES

Permanent failure rates for the WWCS and ARCS computers were predicted by General Electric based on MIL-HDBK-217B. To make a comparison between the WWCS and the ARCS meaningful, the failure rates (previously submitted by GE) for the WWCS, which were based on a different source, had to be adjusted.

Permanent failure rates for sensors and servos were based on airline maintenance data accumulated by Boeing.

Additional details may be found in Table H-2.

3.0 TRANSIENT FAILURE RATES AND LEAKAGES

Four different transient fault effects were considered in the study:

- 1) Electrical Power Transients
- 2) Lightning
- 3) Hydraulic Pressure Transients
- 4) Sensor Nuisance Failures

These different sources are discussed below with the emphasis on Sensor Nuisance Failures which in the past has been the dominating nuisance failure source.

3.1 ELECTRICAL POWER SYSTEM TRANSIENTS

Electrical power system transients may be classified as being either normal or abnormal. Normal transients are transients which can be expected during normal operation of the airplane with the equipment operating normally.

TABLE H-1. SOURCES UTILIZED FOR THE
MARKOV MODEL PARAMETER ASSESSMENT TASK

SOURCE	PARAMETER	PERMANENT FAULT RATES	TRANSIENT FAULT RATES	RATE RATIO	LEAKAGE
MIL-HDBK-217B (COMPUTER)		X			
AIRLINE MAINTENANCE DATA (SENSORS, SERVOS)		X			
ANALYSIS (SENSORS)			X		X
STATISTICAL SAMPLING (SERVOS)				X	
BOEING AND GE EXPERIENCE			X	X	X
LITERATURE INFORMATION			X	X	

TABLE H2
PERMANENT FAULT RATES

COMPUTER (PROCESSOR, MEMORY, I/O) : MIL-HDBK-217B

- COMPONENT QUALITY FACTOR π_Q :

PART TYPE	QUALITY DESIGNATORS
MICROELECTRONICS	A, B, (B-1)*, B-2, C
DISCRETE SEMICONDUCTORS	JANTXV, JANTX, (JAN)
CAPACITORS ESTABLISHED RELIABILITY(ER)	L, M, P, (R), S
RESISTORS ESTABLISHED RELIABILITY(ER)	(M), P, R, S
ALL OTHER PARTS COVERED BY OLDER (NON-ER) SPECIFICATIONS	UPPER, (MIL-SPEC), LOWER

* QUALITY LEVEL B-1 MAY BE ACHIEVED BY TESTING ACCORDING TO
MIL-STD-883, METHOD 5004, CLASS B

- ENVIRONMENTAL FACTOR π_E : AIRBORNE INHABITED

SENSORS AND SERVOS: AIRLINE MAINTENANCE DATA

- NATIONAL AND EASTERN AIRLINES SHOP DATA
FLIGHT HOURS: EASTERN - 1,886,643, NATIONAL - 566,863
CALENDAR PERIOD : 1/70 - 1/75
- UNITED AND LUFTHANSA EQUIPMENT REMOVAL DATA:
FLIGHT HOURS : UNITED - 1,155,525, LUFTHANSA - 274,831
CALENDAR PERIOD: 1/72 - 1/75

3.1 (cont'd)

Abnormal transients are those that do not necessarily occur in the normal operation of the airplane. They are usually the result of equipment malfunction or failure or the use of abnormal operating procedures.

A few of the commonly encountered normal transient sources are listed in Table H-3 below. Table H-4 displays a few abnormal transient sources.

The transient data presented shows that normal transient faults will have a duration in the order of 50 ms. The rate of occurrence is estimated not to

exceed	20	times	per	hour.	With	this	high	occurrence	rate	it	will
become	necessary	to	design	the	hardware,	for	example	DC - power	supplies		
such	that	all	normal	transients	will	be	absorbed	without	degradation		

of the system operation. This approach, which agrees with common avionic system requirements, implies that the leakages associated with normal power transients are zero; only a design error would result in a nonzero leakage.

The most critical abnormal transients are short circuits caused for example by a failed module. Depending on the isolation mechanism, i. e. whether electronic or thermal, it may take several seconds to clear these faults — a period during which the power supply voltage could drop essentially to zero. It is estimated that these abnormal transients will not occur more often than once per one hundred hours, i. e. a fault rate equal to 10^{-2} /hour.

It will be assumed for the ARCS, that each channel is powered from a separate power bus where a channel includes the sensor, computer and servo stages. It will further be assumed that each sensor signal will always recover following a power loss, recognizing that this recovery may include warmup and filter settling time. A servo stage will also be intact following a power loss but will always recover in the servo off state.

Regarding the computer stage, reliance will be put on the ARCS ability to automatically start up and resynchronize following a power loss. This will also apply to the remote situation of a simultaneous power loss in two channels.

TABLE H-3 NORMAL POWER TRANSIENTS (EXAMPLES)*

SOURCE	DURATION	EFFECT
AC-LOAD SWITCHING	50MS	-30% TO +10% BUS VOLTAGE VARIATION
INDUCTIVE SWITCHING	50MS	500V P-P RIPPLE
AC POWER SOURCE TRANSFER	50MS	-80% VOLTAGE DROP
DC-LOAD SWITCHING	---	1-2V DC-BUS VOLTAGE DROP (28V)

TABLE H-4 ABNORMAL POWER TRANSIENTS (EXAMPLES)*

SOURCE	DURATION	EFFECT
SHORT CIRCUIT	150MS-6SEC	TEMPORARY POWER LOSS
POWER GENERATING EQUIPMENT FAILURE OR MALFUNCTION	4 SEC	LOSS OF EXCITATION OR OPEN CIRCUIT
OUT-OF-PHASE GENERATOR CONNECTION	100MS	LARGE VOLTAGE DROP

*EXTRACTED FROM: "AIRCRAFT ELECTRICAL POWER SYSTEM TRANSIENTS"
MARLYN B. WALL
AVION. MAINT. CONF. SAN FRANCISCO 1973

3.1.5.2 (cont'd)

(This event has sufficiently low probability that the ARCS system would be acceptable even without this capability. It is however "for free" since it is required as a part of ARCS initial automatic start up function.)

If the restart function is properly designed, the system would, at least in principle, be able to recover from all possible transient power loss events. In the ARCS trade and reliability study it is therefore assumed that the computer transient power fault leakage is zero (nominally).

3.2 LIGHTNING STROKES

There are generally two separate time phases of a lightning stroke (figure H-1). One short initial phase with a duration in the order of a few tens of micro-seconds during which the current may reach a value of $2 \cdot 10^5$ A, and a second phase with a duration up to a second and current levels in the order of a few hundred amperes. This latter phase is the more critical from structural damage viewpoint while the short initial phase may give rise to potentially harmful electromagnetic interference. Experience and tests show that lightning induced power bus transients usually will be of a magnitude not exceeding 600 volts peak-to-peak and a duration in the order of 50 μ s (figure H-2). Lightning induced transients, therefore, have an effect similar to transients caused by inductive switching, with the difference that they may occur anywhere in the system, in particular in longer wire connections as for example to the sensors. The effect of these transients on analog hardware may be suppressed by proper design methodology. The effect on digital hardware, for example whether an induced transient could scramble the memory, is more difficult to assess but experience indicates that this possibility appears remote. Aircrafts equipped with digital Inertial Navigation Systems have, for example, on several occasions been subjected to lightning strokes without any noticeable effect on digital system operation.

Since it has not been shown that lightning has a significant effect on digital system operation, the transient leakage due to lightning will, nominally, be set equal to zero in the ARCS reliability analysis.

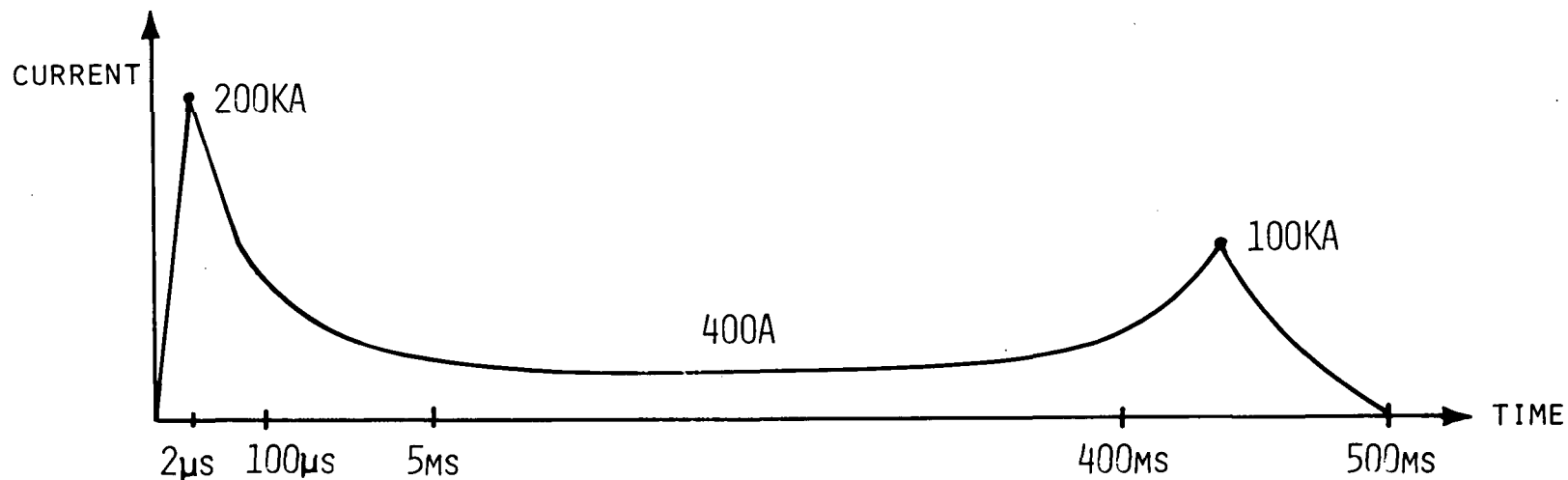
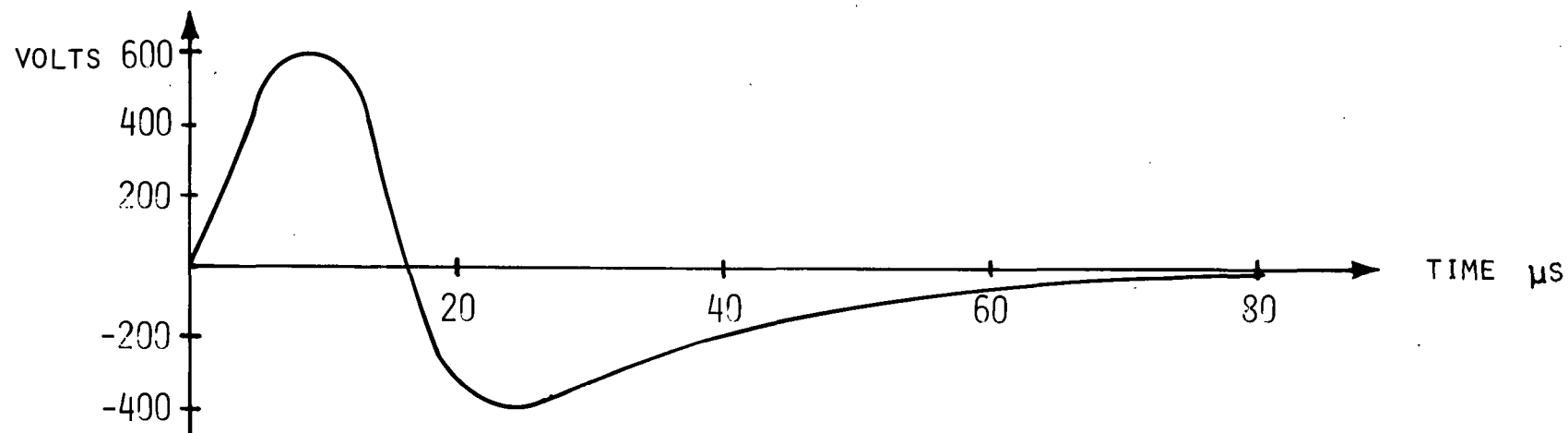


FIGURE H-1 CHARACTERISTIC WAVEFORM OF LIGHTNING STROKE*



* "SPACE SHUTTLE LIGHTNING CRITERIA DOCUMENT"
JSC-0636 JOHNSON SPACE CENTER SEPT. 1973

** "AIRCRAFT ELECTRICAL POWER TRANSIENTS"
MARLYN B. WALL AVION. MAINT. CONF. SAN FRANCISCO 1973

FIGURE H-2. CHARACTERISTIC ELECTRICAL TRANSIENT INDUCED BY LIGHTNING**

3.3 HYDRAULIC POWER TRANSIENTS

The most common hydraulic pressure transients are shock waves caused by sudden valve closure and pressure droops caused by a sudden massive demand. However, these effects should, according to Boeing expertise, be manageable by proper worst case design so that hydraulic power system pressure variations will always stay within $\pm 50\%$ of the nominal line pressure. Thus, if high and low limit pressure switches were set at those values, transient free operation would result.

For the ARCS trade study and reliability evaluation, it is therefore assumed that transient hydraulic power faults nominally have zero rates.

3.4 SENSOR NUISANCE FAULTS

Sensor nuisance faults are usually caused by temporary disagreements between redundant sensor signals. These sensor disagreements could be caused by "static" effects like bias or scalefactor differences or they could be dynamic in nature, stemming from different sensor dynamics or differing inputs. Examples of dynamic disagreements caused by deviating inputs are accelerometers sensing different accelerations due to mounting structure vibrations or air data signals sensing the air pressure in different pitot tubes.

Static deviations and dynamic deviations caused by unmatched dynamic sensor responses may, at least in theory, be eliminated by improved sensor design or compensation algorithms. One such algorithm developed by Boeing is the Compensated Limited Average (CLA) algorithm (see reference H-1) which compensates for bias deviations between analog sensor signals. Other, more sophisticated algorithms, which perform bias as well as scalefactor compensation, have been designed and presently are under evaluation by the Flight Controls Technology Staff at Boeing.

3.4 (cont'd)

By subtracting off the static effects from the sensor difference signal, a compensated (or equalized) signal results, which more closely represents the dynamic part of the signal deviation. This dynamic error signal may be considered random, exhibiting certain statistical properties like variance and spectrum. The random signal is monitored by a dynamic failure detection mechanism which ideally should be able to distinguish between sensor disagreements due to randomness and "real" sensor failures. The detection mechanism used by ARCS consists of two fixed thresholds, one positive and one negative. A fault is detected whenever a difference signal exceeds the preset thresholds.

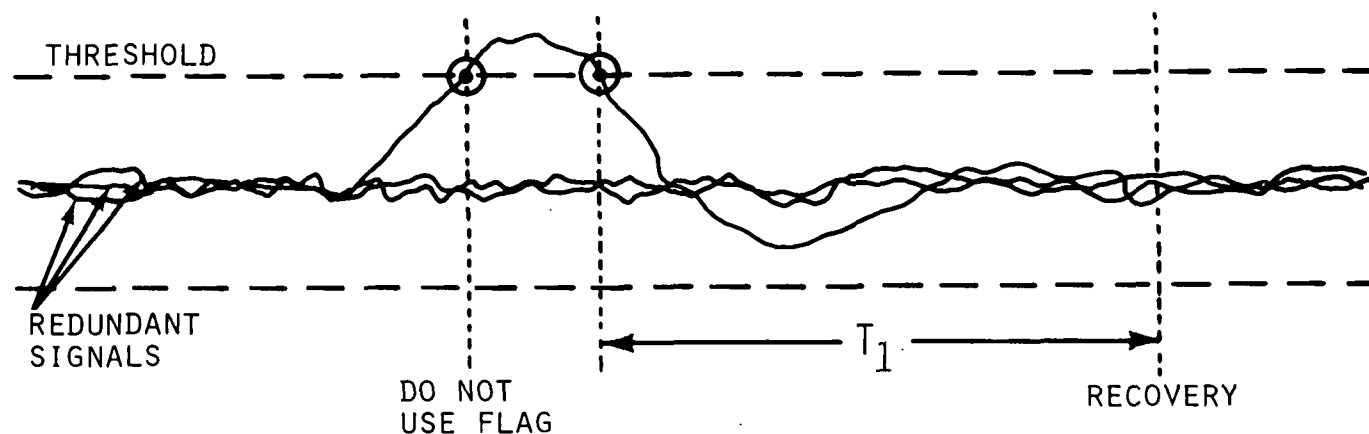
3.4.1 Faults a Triplex or Higher Redundancy

The ARCS strategy for sensor faults occurring at triplex (quadruplex) redundancy level depicted in figure H-3 consists of reconfiguring from triplex (quad) to duplex (triplex) immediately upon a threshold exceedance. The deviating signal will not be used unless it recovers at some later time within the threshold band and remains within this band a prescribed time period T_1 .

The recovery of a previously faulty signal may take place at any time after the occurrence of the fault until a time when a second like fault occurs. A second fault in a like sensor will immediately cause a first fault to be declared permanent. As a consequence, permanent hardover failures will not be classified as such until a second fault occurs.

The strategy described will exhibit outstanding nuisance fault rejection capability. However, there are two situations which have to be analyzed.

The first is the possibility that a second fault occurs before the first fault has time to recover, i. e., before the end of the T_1 period. This would lead to the first fault being declared permanent. A transient leakage would then take place in case the first fault was caused by a transient condition.



STRATEGY:

THE REDUNDANCY LEVEL IS TEMPORARILY DEGRADED FOLLOWING A TRANSIENT DISAGREEMENT. THE DEVIATING SIGNAL WILL NOT BE USED UNLESS IT RECOVERS AND REMAINS WITHIN THRESHOLD FOR A PERIOD T_1 . AT SOME LATER TIME, IN CASE A SECOND DISAGREEMENT BETWEEN THE REMAINING SIGNALS OCCUR WITHIN THE INTERVAL T_1 THE FIRST TRANSIENT FAULT WILL BE DECLARED PERMANENT. THIS STRATEGY PROVIDES PROTECTION FROM OSCILLATORY FAILURES. T_1 IS DETERMINED BY THE RISK OF ENCOUNTERING ANOTHER (LIKE) FAULT BEFORE T_1 HAS ELAPSED SINCE THIS WOULD RESULT IN A LEAKY TRANSIENT.

FOR THE ARCS STUDY : $T_1 = 10s$

FIGURE H-3 SENSOR REDUNDANCY MANAGEMENT STRATEGY AT TRIPLE (OR QUAD) REDUNDANCY

The second situation to be analyzed is the possibility and consequence of a latent failure caused by a false recovery. A signal, for example a rate gyro output, failing to zero could falsely recover at some later time of when the good signal is not zero. A subsequent failure to zero in a redundant rate gyro could cause an undetected system failure. Both considerations above will influence the selection of T_1 , the first in the direction of a shorter and the second in the direction of a longer T_1 period.

The first fault leakage, ℓ , will be given by the probability of a second fault occurring in the T_1 interval and may be expressed by:

$$\ell = N \cdot (\lambda_p + \lambda_T) \cdot T_1$$

with $N = 2$ or $N = 3$ for a triplex and a quad stage respectively and with λ_p the permanent and λ_T the transient fault rate.

The situation of latent failures and the possibility of false recoveries, i.e., recovery of a failed signal, will now be addressed.

A latent failure occurs when a signal fails passively in a way such that it is undetected by available signal monitors. Usually a latent failure will be detected at some later time. A sensor signal failing passively will for example be detected when the good signals are caused to deviate significantly from the passively failed signal. On the other hand, as discussed above, a detected passive failure may at some later time falsely recover. These two situations may be modelled by the Markov representation of figure H-4. of figure is partitioned into two states, 3a and 3b,

State 1 in the diagram is the unfailed state, state 2 the latent failure state, state 3a the detected but not isolated passive failure state and state 3b the detected and isolated failure state. f_1 and f_2 are fractions representing the conditional probability of an undetected failure and the conditional probability of a detected but not isolated failure, given a failure.

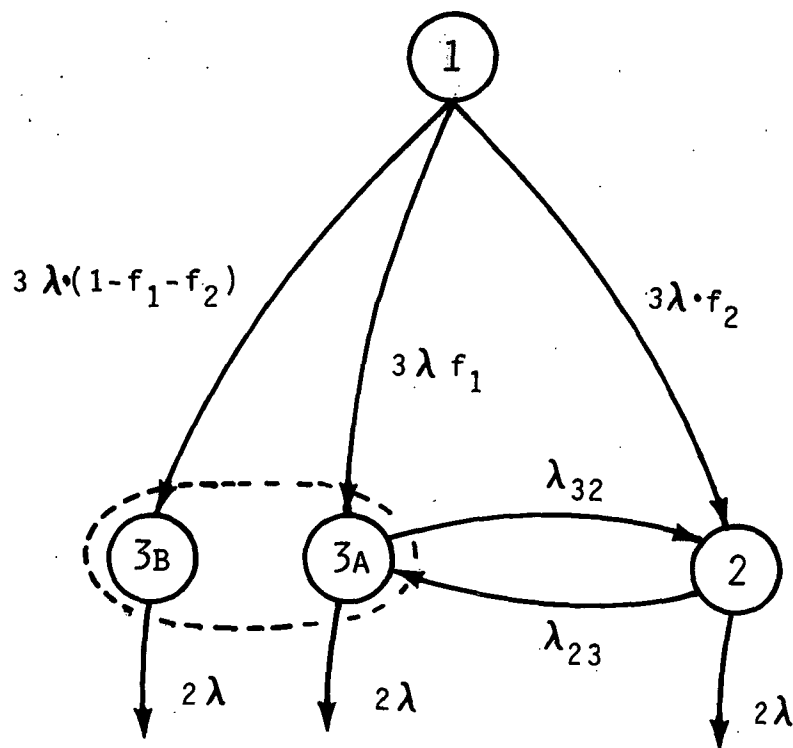


FIGURE H-4

3.4.1 (cont'd)

λ_{23} is the rate of detecting a latent failure and λ_{32} the rate of false recovery of a passively failed signal. λ_{23} may be assessed by estimating the threshold exceedance rate of the difference between an active signal and a passively failed signal. The rate λ_{32} may be assessed as follows:

Let $\lambda_{23} = \lambda$ be the latent failure detection rate and assume quiescent flight conditions with occasional exceedances as depicted in figure H-5. Let the time between two exceedances be t_d . A passively failed signal will falsely recover if $t_d > T_1$ where T_1 is the SSFD recovery time of figure H-3.

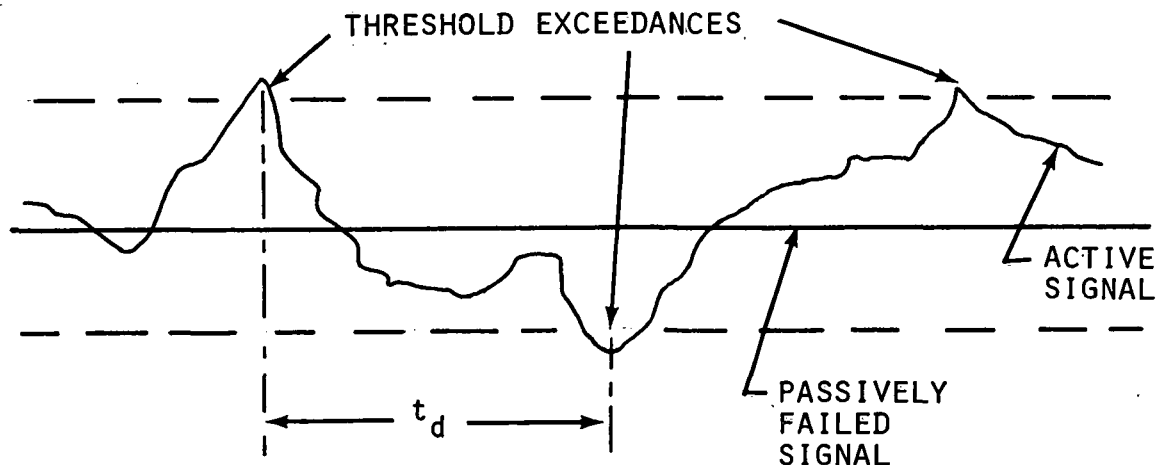


FIGURE H-5 LATENT FAILURE SITUATION

The probability of no threshold exceedance in the time interval $[0, T_1]$, i. e., the probability of a latent recovery is, assuming a Poisson distribution:

$$p(\text{latent recovery}) = \exp(-\lambda T_1)$$

Furthermore, the probability of exactly n exceedances in a time interval $[0, t]$ is given by:

$$p[\text{number of exceedances equal to } n] = \frac{(\lambda t)^n}{n!} \exp(-\lambda t)$$

The probability that a latent signal will not falsely recover in the time interval $[0, t]$ is then given by:

$$\begin{aligned} p(\text{no false recovery}) &= \sum_{n=0}^{\infty} \frac{(\lambda t)^n}{n!} \exp(-\lambda t) \cdot [1 - \exp(-\lambda T_1)]^n = \\ &= \exp[\lambda \cdot (1 - \exp(-\lambda T_1)) \cdot t] \cdot \exp(-\lambda t) = \\ &= \exp(-\lambda \exp(-\lambda T_1) \cdot t) \end{aligned}$$

This probability is the solution of

$$\dot{p} = -\lambda \exp(-\lambda T_1) \cdot p; \quad p(0) = 1$$

which means that

$$\lambda_{32} = \lambda \cdot \exp(-\lambda T_1) = \lambda_{23} \exp(-\lambda_{23} T_1)$$

Since the CARSRA program will only handle unidirectional transitions, the failure rates λ_{23} and λ_{32} have to be condensed into an equivalent rate λ_{23}^* . This can be done as follows: the probability mass flowing from state 3 to 2 in a small time element dt is

3.4.1 (cont'd)

$$dp_{32} = \lambda_{32} \cdot p_{3a} \cdot dt \approx \lambda \cdot e^{-\lambda T_1} \cdot 3\lambda_p f_1 t \cdot dt$$

(It can be shown that $p_{3a} \approx 3\lambda_p f_1 t$ and

$$p_2 \approx 3\lambda_p f_2 t \text{ where } \lambda_p \text{ is the module failure rate})$$

The incremental mass flowing from state 2 to 3 is

$$dp_{23} = \lambda_{23} \cdot p_2 \cdot dt = \lambda \cdot f_2 \cdot 3\lambda_p t \cdot dt$$

so that

$$\begin{aligned} dp_{23} - dp_{32} &= \lambda \cdot f_2 \cdot 3\lambda_p t \cdot dt - \lambda \cdot e^{-\lambda T_1} \cdot 3\lambda_p f_1 t \cdot dt = \\ &= \left(\lambda - \lambda \cdot e^{-\lambda T_1} \cdot \left(\frac{f_1}{f_2} \right) \right) \cdot 3\lambda_p f_2 t \cdot dt = \\ &= \lambda \left(1 - e^{-\lambda T_1} \cdot \left(\frac{f_1}{f_2} \right) \right) p_2 \cdot dt \end{aligned}$$

The equivalent rate λ_{23}^* then becomes

$$\lambda_{23}^* = \lambda \left(1 - e^{-\lambda T_1} \cdot \left(\frac{f_1}{f_2} \right) \right)$$

which could be positive or negative. In the case of a negative λ_{23}^* , the appropriate model will be using an equivalent rate λ_{32}^* derived as follows:

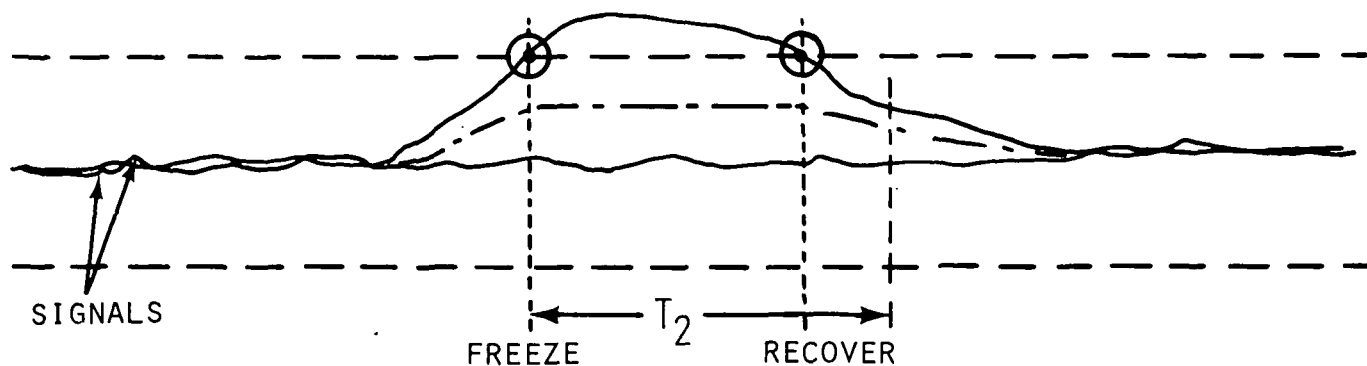
$$dp_{32} - dp_{23} = \lambda \left(e^{-\lambda T_1} \cdot \left(\frac{f_1}{f_2} \right) - 1 \right) \cdot 3\lambda_p f_2 t \cdot dt =$$

$$= \lambda \left(e^{-\lambda T_1} \cdot f_1 - f_2 \right) (1 - f_2)^{-1} p_3$$

$$\therefore \lambda_{32}^* = \lambda \left(e^{-\lambda T_1} \cdot f_1 - f_2 \right) (1 - f_2)^{-1}$$

Faults at Duplex Redundancy

The ARCS strategy for handling sensor transient faults when operating in duplex will be different from the strategy employed when operating at a triplex or quadruplex redundancy level. Upon a disagreement between two remaining operational sensor signals, the selected signal will be held, or frozen, at the value it had at the point of the disagreement (see figure H-6). This applies to failures occurring in any of the different stages in the ARCS. The frozen condition will impair the control of the aircraft to an extent depending on the flight mode and the location and the duration of the fault. The automatic landing mode will for example be more critical than a cruise mode (CWS), and servo command faults more critical than most sensor faults. The maximum permissible duration, before the failure condition must be resolved, will be determined by several different factors including the particular control law implementation and the dynamic characteristics of the controlled aircraft, and will eventually, in a practical implementation, have to be determined by closed loop simulation. However, performing a closed loop simulation is a major task which falls outside the scope of the ARCS program. The ARCS reliability and trade study therefore used already available data from a study performed on the 747 automatic landing system (reference H-2). In this study, which was performed to support the certification of the 747 autoland system, the aircraft trajectory responses to a number of fault conditions were studied using a closed loop simulation. A failure



STRATEGY:

THE SELECTED OUTPUT SIGNAL IS FROZEN AT THE INSTANT OF DISAGREEMENT AND IS RECOVERED IMMEDIATELY UPON RETURN WITHIN THE THRESHOLD PROVIDED THIS OCCURS WITHIN THE INTERVAL T_2 .

IF THE SIGNAL DOES NOT RETURN WITHIN THE T_2 - INTERVAL, AND THE FAULT HAS NOT BEEN ISOLATED, THE STAGE IS FAILED.

FOR THE ARCS STUDY: $T_2 = 1$ SECOND

FIGURE H-6 SENSOR REDUNDANCY MANAGEMENT AT DUPLEX REDUNDANCY

annunciation delay of less than or equal to one second was found to be acceptable for all faults considered. One second will therefore conservatively be selected as being the maximum recovery time for ARCS faults occurring at a duplex stage redundancy level, regardless of the location of this fault.

Upon a disagreement between two remaining signals, recovery will thus be attempted up to one second after the occurrence of the fault. If recovery is not successful in this time period, a passive system failure will be announced for the near and intermediate application models. A possible strategy for the far term Fly-By-Wire application could be to randomly select one of the remaining signals. In the analysis of this system it was however assumed that the system fails upon an unresolved disagreement between two remaining signals.

3.5

ANALYTICAL TOOLS FOR SENSOR NUISANCE FAILURE DETECTION

In order to be able to predict reliability model parameters like transient fault rates and leakages, analytical expressions are needed for threshold exceedance rates as well as the probability distribution of the threshold exceedance duration. The following formula for the positive or negative threshold exceedance rate in exceedances per second, assuming a Gaussian process, is given in reference H-3 .

$$\text{Threshold Exceedance Rate} = \text{TER} = \frac{1}{2\pi} \left(\frac{\lambda_2}{\lambda_0} \right)^{\frac{1}{2}} e^{-\frac{u^2}{2\lambda_0}} \text{ per second}$$

$$\text{where } \lambda_0 = r(0); \quad \lambda_2 = \left[\frac{d^2 r}{dt^2} \right]_{t=0}$$

$$u = \text{normalized threshold} = \frac{\text{threshold}}{\text{std. dev.}}$$

and $r(t)$ is the autocorrelation function. In reference H-4 this formula has been generalized to non-Gaussian processes under the assumption of independency between the amplitude and the derivative of the amplitude. The generalized expression is:

$$\text{TER} = \frac{p(u)}{p(0)} \cdot \frac{1}{2} \text{ ZPS}$$

Where $p(\cdot)$ is the amplitude probability distribution and ZPS is the zero crossing rate, i. e., mean time between zero crossings.

In the case of a second order process, it may be shown that

$$\frac{1}{2\pi f_1} \cdot \left(\frac{\lambda_2}{\lambda_0} \right)^{1/2} = B = \text{bandwidth in cps.}$$

Which suggests the formula:

$$TER = \frac{p(u)}{p(0)} \cdot B$$

This expression will be used in the following. In reference H-3 it is also shown that the average time spent above a certain threshold, u , is given by:

$$\text{Average time above } u; T_A = \frac{1 - P(u)}{TER} = \frac{P(-u)}{TER}$$

$$\text{Where } P(u) = \int_{-\infty}^u p(u) du$$

This expression holds regardless of the symmetric amplitude density $p(\cdot)$.

A useful approximation when $p(u)$ is the Gaussian distribution is:

$$T_A = \frac{|u|}{B(1+u^2) \sqrt{2\pi}}$$

which holds for $|u| > 2$.

An exact expression for the probability distribution of the time, spent over a threshold, appears to be difficult to derive. An approximate expression which holds for large thresholds assuming a Gaussian process is however given in reference H-3:

3.5 (cont'd)

$$\text{Prob (exceedance time } > t) \approx e^{-\frac{\pi}{4} \left(\frac{t}{T_A} \right)^2}$$

In the development below it will be assumed that this formula is good enough to permit the assessment of leakages at duplex redundancy.

A more compact formula for the nuisance failure rate at duplex redundancy may be obtained by observing that a nuisance failure results when (a) the threshold is exceeded and (b) the signal stays above the threshold too long. This may be expressed

$$\begin{aligned} \text{NFR} &= \text{Nuisance Failure Rate} = \\ &= 4 \times \text{TER} \times \exp \left(-\frac{\pi}{4} \left(\frac{T_2}{T_A} \right)^2 \right) \end{aligned}$$

where T_2 is the maximum permissible recovery time in duplex. Substituting the expression for T_A :

$$\text{NFR} = 4 \times \text{TER} \times \exp \left(-\frac{\pi}{4} \left[\frac{\text{TER} \cdot T_2}{P(-u)} \right]^2 \right)$$

This expression may be maximized with respect to TER so that:

$$\text{NFR} \leq \text{NFR}_{\text{max}} = 4 \times \sqrt{\frac{2}{\pi}} \times e^{-1/2} \times \frac{P(-u)}{T_2} < \frac{2 \times P(-u)}{T_2}$$

This simple expression may be used to conservatively estimate the nuisance failure rate at duplex redundancy. Observe that the expression does not depend on the bandwidth, B, of the signal!

3.6

AVAILABLE SENSOR STATISTICS

Sensor signal statistics have been recorded in several flights with an experimental 737 aircraft (the NASA 515 test vehicle). The recorded data, which consists of raw digitized signals from triplicated flight control sensors, has been processed by Boeing to extract information like mean values, variances, power spectra, histograms, etc. for signal amplitudes and/or signal differences. This statistical data base was used in the ARCS Reliability/Trade Study to assess sensor transient fault rates and leakages.

Table H-5 shows standard deviations for a number of sensor signal differences, i. e., the difference between two like signals. The data is presented for both quiescent conditions and rough turbulent conditions.

For this study the assumption will be made that the dynamic sensor deviation signals are Gaussian distributed with a process bandwidth of 1 cps. The Gaussian assumption is adopted since insufficient information presently is available about the detailed structures of the tails of the density functions of interest. The 1 cps bandwidth assumption is consistent with available power spectral density plots for rough turbulent conditions.

3.7

ICPS SENSOR TRANSIENT FAULT RATE AND LEAKAGE ANALYSIS

The Incremental Control Processor System (ICPS), currently installed on the NASA 515 experimental aircraft has, in the past, been plagued by sensor nuisance failures. To test the validity of the developed analysis tools, the nuisance failure rates for this system was predicted assuming the SSFD algorithm documented in reference H-5.

Random fault rates will first be addressed under the assumption that deterministic errors caused by bias and scalefactor influences have been removed by proper compensation. The effects of deterministic errors will be addressed separately later.

The ICPS SSFD design consists of two cascaded first order filters followed by threshold detectors and timeout counters (figure H-7). Inputs to

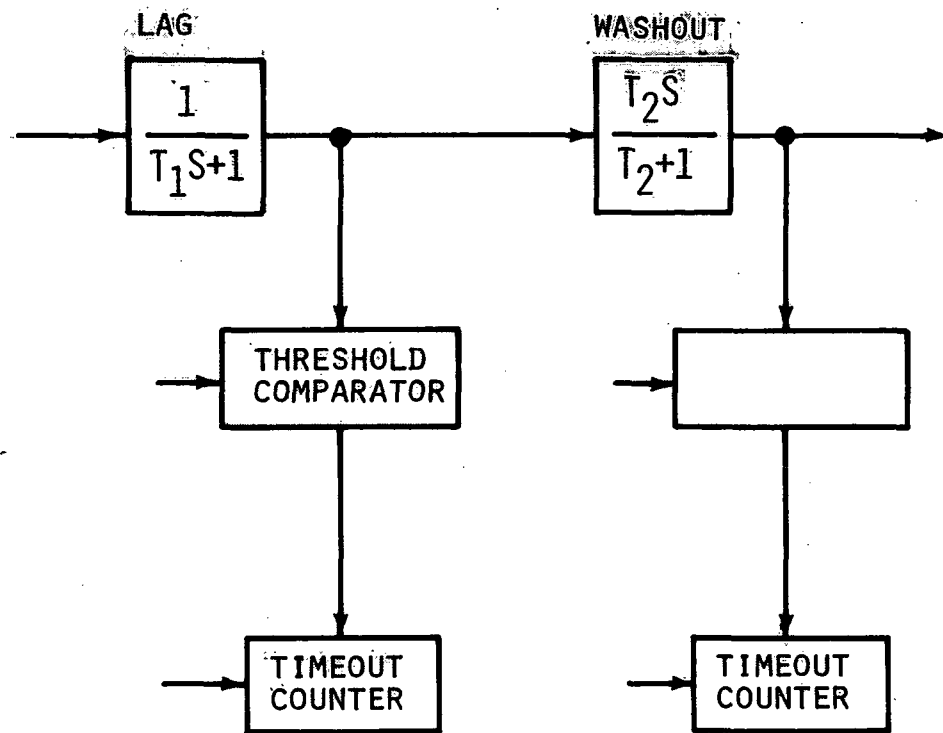


FIGURE H-7 ICPS SSFD ALGORITHM

these filters are the signal differences between individual sensor signals, i.e., A-B, A-C, and B-C.

Table H-6 shows the threshold values and delays initially selected for the ICPS which are documented in reference H-5. . The numbers of delay counts have been translated to seconds using 6.144 ms per count.

To be able to use the formulas for threshold exceedance rates and average time above a threshold, which were derived in section 3.5, the standard deviations and the bandwidths of the filtered difference signals have to be estimated. Standard deviations may be assessed by using the formulas:

$$\begin{aligned}\sigma_o^2 &= \int_0^\infty |G(\omega)|^2 |x(\omega)|^2 d\omega \\ \sigma_I^2 &= \int_0^\infty |x(\omega)|^2 d\omega\end{aligned}$$

Where σ_o^2 is the variance of the filter output, σ_I^2 the variance of the filter input, $x(\omega)$ the fourier transform of the input signal and $G(\omega)$ the filter transfer function with $s = j\omega$. For simplicity it will be assumed that the input spectrum, $|x(\omega)|^2$, is flat in an interval $[0, 2\pi f]$ with $f = 1$ cps and vanishes for higher frequencies.

With this assumption, $\sigma_I^2 = \text{variance in Table H-5} = 2\pi \cdot 1 \cdot |x(\omega)|^2$ so that:

$$\begin{aligned}|x(\omega)|^2 &= \frac{\sigma_I^2}{2\pi} & 0 < \omega \leq 2\pi \\ |x(\omega)|^2 &= 0 & \omega > 2\pi\end{aligned}$$

Furthermore for the lag filter:

$$G(\omega) = \frac{1}{j\omega T_1 + 1}$$

$$\sigma_{oL}^2 \approx \left(\int_0^\infty \frac{1}{(\omega T_1)^2 + 1} d\omega \right) \cdot \frac{\sigma_I^2}{2\pi} =$$

ORIGINAL PAGE IS
OF POOR QUALITY

TABLE H-6 ICPS SS/FD THRESHOLDS AND TIME DELAYS

SENSOR SIGNAL	TH1	T1(sec)	TD1 (sec)	TH2	T2 (sec)	TD2 (sec)
\ddot{h}	0.47 FPS ²	3.14	0.025	0.23 FPS ²	12.6	0.340
\dot{h}	5.6 FPS	12.6	0.025	1.25 FPS	6.3	0.340
θ	2°	12.6	0.025	0.47°	6.3	0.340
$\dot{\theta}$	0.5°/s	12.6	0.025	0.47°/s	6.3	0.340
ϕ	7.0°	0.78	0.098	2.54°	1.6	0.098
$\dot{\phi}$	1.95°/s	3.14	0.098	1.87°/s	3.14	0.098
$h_{R/A}$	100 FT	1.57	0.025	100 FT	0.39	0.340
GSE	0.100°	12.6	0.025	0.023°	6.3	0.340
LOC	0.293°	1.57	0.098	0.195°	1.57	0.098
F_{COL}	3.38 LB	1.57	0.025	16.54 LB	0.39	0.340
F_{WHL}	3.38 LB	1.57	0.025	16.54 LB	0.39	0.340

3.7 (cont'd)

$$= \frac{\sigma_I^2}{2\pi T_1} \int_0^{\infty} \frac{1}{x^2+1} dx = \frac{\sigma_I^2}{2\pi T_1} (\text{ATAN}(\infty) - \text{ATAN}(0))$$

$$\underline{\sigma_{oL} = \frac{\sigma_I}{2\sqrt{T_1}}}$$

Where the assumption has been made that $T_1 \gg 0.25$, so that the contribution to the integral for frequencies above 1 cps is negligible.

For the lag - washout combination we get:

$$G(\omega) = \frac{1}{(j\omega T_1 + 1)} \cdot \frac{j\omega T_2}{(j\omega T_2 + 1)}$$

$$\text{and } \sigma_{o\omega}^2 = \frac{\sigma_I^2}{2\pi} \int_0^{\infty} \frac{(\omega T_2)^2}{[(\omega T_1)^2 + 1][(\omega T_2)^2 + 1]} d\omega$$

which yields:

$$\sigma_{o\omega} = \frac{\sigma_I}{2\sqrt{T_1 \left(1 + \frac{T_1}{T_2}\right)}}$$

The bandwidth of the filtered signal is for the lag filter approximated by:

$$f_{BL} = \frac{1}{4T_1} \text{ cps}$$

For the lag - washout combination:

$$f_{BW} = \max \left[\frac{1}{4T_1} ; \frac{1}{4T_2} \right] \text{ cps}$$

Using these relations and the previously derived formulas for Threshold Exceedance Rate, TER, the Average Time above the threshold, T_A , and the probability distribution for time spent above a threshold, Table H-7 may be constructed. The data in Table H-7 applies to rough turbulent conditions. Contributions to transient fault rates from quiescent parts of the flight will be negligible since the rates increase faster than exponential with increasing standard deviations. The data of Table H-7 should therefore be modified by multiplication by the fraction of the total flight time spent in turbulent conditions.

It appears that the vertical velocity and perhaps even the roll angle signal will cause nuisance failure problems. This prediction has been verified by flight testing which has indicated that the SSSFD mechanization with the parameters of Table H-6 will lead to excessive nuisance failure rates. However, the nuisance failure problems may be alleviated by opening up the washout filter thresholds for the vertical velocity and roll angle signals.

Deterministic Errors: Deterministic errors are signal deviations that may be eliminated by compensation. Two main sources are sensor signal bias and scale factor differences. Table H-8 displays biases and scale factors for a selected number of sensor signal deviations where the deviation of a certain signal, ΔA , is the difference between this signal and the mean value of three signals, i. e.

$$\Delta A = A - \frac{1}{3} (A + B + C)$$

except for the track angle, ΔTKA , and localizer signals for which the deviation is derived from two signals:

$$\Delta A = A - \frac{1}{2} (A + B)$$

TABLE H-7 PREDICTED ICPS TRANSIENT SENSOR FAILURE DATA IN TURBULENT FLIGHT

LAG FILTER						LAG AND WASHOUT FILTER					
SENSOR SIGNAL	σ_{OL} lag output	TER (PER HOUR)	T_A SEC	LEAKAGE *	TRANSIENT FAILURE RATE/HOUR	σ_{OH} wash output	TER PER HOUR	T_A SEC	LEAKAGE *	TRANSIENT FAILURE RATE/HOUR	
\ddot{h}	.045 FPS ²	3×10^{-21}	0.504	0.998	3×10^{-21}	.040 FPS ²	1.5×10^{-4}	0.856	0.88	1.3×10^{-4}	
\dot{h}	.916 FPS	3×10^{-6}	0.63	0.99	3×10^{-6}	.929 FPS	0	0.22	0.99	0	
θ	.034°	0	—	—	0	.0195°	0	—	—	—	
$\dot{\theta}$	2.8×10^{-3} °/s	0	—	—	0	1.6×10^{-3} °/s	0	—	—	—	
ϕ	0.96°	2×10^{-8}	0.168	0.765	1.5×10^{-8}	0.554°/s	2×10^{-1}	0.260	0.89	2×10^{-1}	
$\dot{\phi}$	0.02 °/s	0	0.004	0.016	0	0.011 °/s	0	0.078	0.003	0	
$h_{R/A}$	1.63 FT	0	—	—	0	0.81 FT	0	—	—	—	
GSE	5.6×10^{-3}	0	—	—	0	3×10^{-4} FT	0	—	—	—	
LOC	8×10^{-3}	0	—	—	0	5.6×10^{-3}	0	—	—	—	
F _{COL}	4.4×10^{-2} LB	0	0.984	0.927	0	2×10^{-2} LB	0	0.121	0.235	0	
F _{WHL}	2.8×10^{-2} LB	0	—	—	0	1.2×10^{-2} LB	0	—	—	—	

FACTORS HAVE BEEN ESTIMATED FROM 1000

* The leakage λ is the conditional probability of a failure indication, given a threshold exceedance

TABLE H-8 SENSOR BIASES AND SCALE FACTORS

SCALE FACTORS HAVE BEEN ESTIMATED FROM 1000 SAMPLES

"Page missing from available version"

The data in the table is based on flight test records. 1000 samples spaced 50 msec in time were used for the estimation of biases and scalefactors during a period of aircraft maneuvering. The correlation coefficient of the last column pertains to the correlation between a (bias compensated) signal difference and the average signal level. A value close to one indicates a significant scalefactor effect.

The Compensated Limited Average (CLA) algorithm is designed to effectively remove bias errors but will not compensate for scalefactor errors. More advanced algorithms have, however, been designed and tested by Boeing which perform on-line scalefactor compensation as well as bias compensation. Sensors exhibiting significant scalefactor errors are the pitch and roll angle gyros for which the scalefactor deviations are in the order of 8-10%. Since the pitch angle threshold level initially was set at 1° and the roll threshold at 3.5° in the ICPS - algorithm, a pitch angle of $1/0.08 = 12^\circ$ or a roll angle of $3.5^\circ/0.1 = 35^\circ$ will cause a threshold exceedance if no scalefactor compensation is employed.

For the study, it will therefore be assumed that effective scalefactor compensation is being employed for these two signals so that the only remaining nuisance failure sources are random in nature.

3.8 ARCS TRANSIENT FAILURE RATE AND LEAKAGE PREDICTIONS

The primary approach to sensor failure monitoring suggested for ARCS is the Compensated Limited Average (CLA) algorithm augmented by scalefactor compensation. The main components of this algorithm is a static detector for slow ramp (or bias) failures and a dynamic detector for oscillatory and hardover failures. The static detector monitors the signal bias level by comparison with prescribed thresholds. The dynamic detector, consists of a threshold comparison followed by a time delay. The strategies of handling faults occurring at duplex and higher redundancy levels were defined in sections 3.4.1 and 3.4.2.

In the Compensated Limited Average (CLA) implementation, the static threshold is set at a level at which a permanently deviating signal adversely would affect the control of the aircraft. This level, which may be established by simulation or engineering judgement, is assumed to be represented by the TH1 values of Table H-6 .

The bias integrating filter will, for this analysis, be treated as a first order lag with a time constant selected to achieve an acceptably low nuisance failure rate. No detection time delay will be implemented following the static failure detector.

The dynamic detector, which monitors the raw unfiltered difference signal, will also be assigned a level compatible with an acceptable nuisance failure rate. The main function of this detector is to provide rapid hardover failure detection.

Analysis indicates that all signals listed in Table H-5 except the three previously troublesome signals, i.e., vertical acceleration, vertical rate and roll angle, may be monitored by setting the static and dynamic thresholds corresponding to the TH1- levels of Table H-6 . Note, however, that the difference signals monitored by the CLA are the deviations between the individual signals and their average value. This implies that the thresholds defined in Table H-6 have to be reduced by one-half for the CLA algorithm. Those threshold levels will, with an integrating time constant of 10-15 sec, define acceptable dynamic failure detection algorithms for all but the above mentioned troublesome signals.

Regarding the vertical velocity signal, which is the most problematic, a dynamic threshold at $TH1 = 5.6/2 = 2.8$ fps will result in an excessive rate. However, monitoring at this level may instead be performed by the static threshold, setting the bias compensating time constant equal to 10 seconds. Hardover failures are detected and isolated without delay by a dynamic detector located at five sigma i.e., $5 \times 6.5/2$ fps = 16 fps. The recovery delay period for

faults at triplex or quadruplex redundancy discussed in section 3.4.1 will be set equal to 10 sec and the threshold exceedance delay for a fault occurring in duplex set equal to one second in agreement with the discussion of section 3.4.2.

It is assumed that the vertical acceleration signal is monitored by a static detector with $TH1 = 0.23 \text{ fps}^2$ and a time constant equal to 5 sec. The dynamic threshold is located at five sigma times the turbulent rms level, i.e., at $5 \times 0.08 = .4 \text{ fps}$.

Finally, it is assumed that the roll angle signal, like the vertical acceleration, is monitored at $TH1 = 3.5^\circ$ by the static detector using a time delay of 5 seconds. The dynamic threshold is set at five times the turbulent rms level, i.e., at $5 \times 0.85^\circ = 4.25^\circ$.

The resulting ARCS transient fault rates are listed in Table H-9.

Note that the data pertains to turbulent conditions. The actual experienced exceedance rates probably will be significantly lower. In the ARCS analysis the assumption was made that the nuisance rates will be ten times lower than in Table H-9.

Initially, the analysis of the ICPS sensor nuisance failures was performed with the intention of using the results for the WWCS reliability analysis. The predicted nuisance failure rates would, however, completely dominate the results. To make the comparison between the ARCS and WWCS meaningful, it was therefore assumed that the nuisance failure rates are the same for these systems.

TABLE H-9 ARCS PREDICTED TRANSIENT SENSOR FAILURE DATA IN TURBULENCE

DIFFERENCE SIGNAL	STATIC DETECTOR						DYNAMIC DETECTOR			
	THRESHOLD	TIME- CONSTANT	OUTPUT STD. DEV	T _{ER} (PER HOUR)	TRIPLEX TRANSIENT LEAKAGE	DUPLEX TRANSIENT LEAKAGE	THRESHOLD	T _{ER} (PER HOUR)	TRIPLEX TRANSIENT LEAKAGE	DUPLEX TRANSIENT LEAKAGE
\ddot{h}	.24 FPS ²	5s	.018 FPS ²	0	~ 0	1	.4 FPS ²	3×10^{-2}	5×10^{-4}	~ 0
\dot{h}	2.8 FPS	10s	.51 FPS	5×10^{-5}	~ 0	1	16 FPS	3×10^{-2}	5×10^{-4}	~ 0
ϕ	3.5°	5s	0.19°	~ 0	~ 0	1	4.25°	3×10^{-2}	5×10^{-4}	~ 0
ALL OTHER SIGNALS	—	—	—	NEGLIGIBLE	—	—	—	—	—	—

ARCS COVERAGE PARAMETER (RATE RATIO) ASSESSMENT

In this section, coverage parameters for permanent failures will be assessed. As a ground rule for the study, it will be assumed that the coverage of any single module failure which occurs at a triplex or higher redundancy level is unity, i.e., that the system always survives a single failure. This failure could, however, be either detected or undetected (latent). The system will survive a single module failure in duplex operation if and only if the particular failure is covered, i.e., if and only if the failure is detected and isolated and if successful redundancy degradation accomplished. Detection of a faulty condition, which could be transient or permanent, will normally be done via signal comparison. Isolation will be attempted based on reasonableness tests and/or module selftest. When a failed module has been successfully isolated, redundancy degradation will be implemented by continuing operation using the remaining, operational, module.

The most difficult redundancy management task is proper failure isolation. Reasonableness tests, which in scope may range from a simple hardover detection to sophisticated algorithms based on modern systems theoretical methods ("analytical" or "functional" redundancy), may be applied for failure isolation. Another possibility is self-test monitors or subsystem level tests of several modules, for example wraparound input-output test.

In the following, coverage, or rather, permanent failure rate ratio parameters, will be assessed for the different ARCS modules.

4.1 COMPUTER COVERAGE

It was recognized early in the ARCS program that an accurate estimation of the (second failure) computer coverage would be virtually impossible without detailed drawings, specifications and circuit diagrams over the computer. Furthermore, even if this documentation were available, the task of estimating the coverage would be large enough to fall outside the scope of the initial ARCS program effort. The decision was therefore made to postulate a certain coverage value which was deemed to be conservative for the ARCS system. The numbers postulated were 0.95 for the second failure coverage, i. e. 95% of surviving a failure in duplex, and 0.90 for the probability of detecting a failure when operating in simplex. If it could be demonstrated that the reliability requirements would be satisfied assuming these numbers, the potential feasibility of the ARCS system would be established. In the case of a future hardware implementation of the system, these assumed parameter values will have to be demonstrated by testing on the actual hardware, possibly supported by a gate level simulation.

4.2 SERVO STAGE RATE RATIOS

At the time of the reliability study the results from the General Electric coverage study (Appendix G) were not yet available. Tentative values for the servo rate ratios were therefore used in the analysis. The coverage in duplex was assumed equal to 0.95 and the probability of detecting a failure when operating in simplex was assumed to be 0.90. These values were later, by the result of the GE coverage study, shown to be representative although somewhat conservative.

SENSOR RATE RATIOS

Rate ratios for the various sensors were assessed based on several different sources. However, it was very difficult to get firm estimates for sensor rate ratios from these sources. The only sensors for which adequate FMEA study results were available, were the Radio Altimeter (R/A) and the ILS receiver. The rate ratio assessment for the other sensors had to be based on rather fragmentary data supported by engineering judgement. As a general ground rule it was assumed that, except for the R/A and ILS, no special dedicated monitoring circuitry was to be used. The monitoring for each of these sensors was assumed to mainly consist of a reasonableness test capable of isolating hardover failures and to some extent passive failures.

Sources consulted to establish sensor rate ratios are listed below:

- "Fail-Operational Autoland Failure Analysis" Boeing Document No. D6-33219
- Vendor Failure Analyses for the Radio Altimeter and ILS Modules
- Boeing In-House Experience
- United Airlines Maintenance Records
- Sensor Failure Mode Data from Airline Maintenance Experience extracted by Boeing
- "Definition Study for Advanced Fighter Digital Flight Control System" AFFDL-TR-75-59 (Rough Draft)

In addition to extracting information from these sources, a special monitoring approach was developed for the R/A sensor. This sensor is critically needed for the autoland function and exhibits a high coverage sensitivity. It is therefore desirable to obtain a high coverage for this sensor. In addition to achieving this goal, the monitoring algorithm developed demonstrates the feasibility of using "analytical redundancy" to improve sensor coverage.

The method utilizes the redundant information available in the vertical acceleration signal from the Normal Accelerometer, together with the vertical velocity signal from the Air Data Computer, to artificially construct a redundant altimeter signal. This signal is of high enough quality so as to serve as a replacement of an R/A signal if it fails. This implies that a R/A failure at duplex redundancy could be perfectly covered by voting between three signals, one of which is artificially created. Figure H-8 shows an example of the performance of the analytically redundant signal during the final phase of a landing. A detailed description of the method is given in reference H-6.

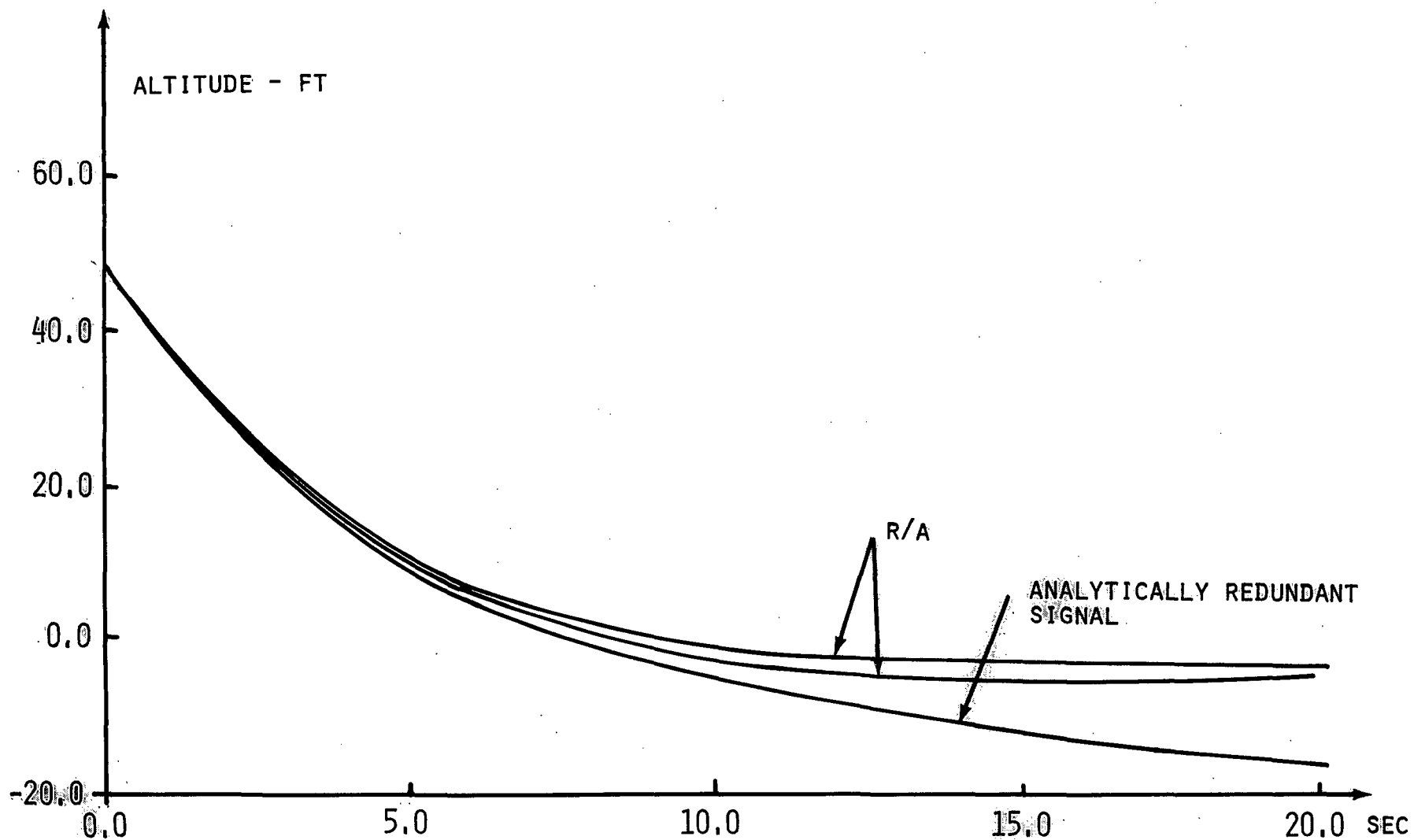


FIGURE H-8 ANALYTICALLY DERIVED ALTIMETER SIGNAL

A summary of the sensor rate ratio parameters used in the study is given in Table H-10. These values are to be considered conservative, except for the R/A and possibly the ILS, in that the coverages, r_{23} , probably could be increased substantially by the use of special hardware monitoring circuits.

TABLE H-10 ASSESSED SENSOR RATE RATIOS

SENSOR	r_{23}	r_{35}
R/A	1.0	0.0
ILS	.99	0.01
YAW RATE	.60	0.75
ACCEL	.75	0.25
DG	.75	0.40
COMPASS	.75	0.40
VG	.75	0.40
CONTR. FORCE	.50	0.50
DADC	.90	0.10
PITOT/STATIC	.50	0.50
ISAD GYRO	.95	0.10

REFERENCES

- H-1 "An Improved CLA Signal Selection and Failure Detection Algorithm for Continuous Signals". Boeing Coord Sheet No. B-8244-10-096.
- H-2 "Fail-Operational Autoland Failure Analysis". Boeing Document D6-33219.
- H-3 "Stationary and Related Stochastic Processes", H. Cramer and M. R. Leadbetter, Wiley 1967.
- H-4 "Statistical Distributions of MTBE Predictions", Boeing Computer Services Coord Sheet No. EM-11.
- H-5 "Redundant Flight-Critical Control System Evaluation", FAA-SS-73-2-2.
- H-6 "Use of State Estimators to Cover Radar Altimeter Failure During Autoland" Boeing Coord Sheet B-8244-10-121.

APPENDIX I

PROCESSOR CROSS-CHANNEL COMMUNICATION LINK STUDY

1.0 INTRODUCTION

The objective of this study was to survey and trade off different methods of implementing the processor cross channel communication function, taking into account the ARCS operational requirements and design concept constraints. The more significant of these are:

- 1) A data transmission rate requirement of at least 20,000 words/s
- 2) A method of implementation which does not significantly impair the effective processor throughput.
- 3) ARCS design criteria and groundrules; in particular the requirement of no interference between computers.
- 4) Redundancy management related requirements and constraints.

Among the many possible cross channel link configurations only a few will satisfy all the ARCS requirements. These configurations are identified and compared with respect to single point failure hazard, reliability, cost and adaptability to optical implementation.

It is shown that the originally proposed ARCS baseline communication link is the implementation that presently best satisfies the requirements.

2.0 POSSIBLE CROSS CHANNEL TRANSMISSION IMPLEMENTATIONS

In the design of a cross channel communication link, several parameters are of interest. The more significant of these are identified below.

2.1 TRANSMISSION TECHNIQUES

The data exchange on a bus is either unidirectional or bidirectional. Furthermore, on a bidirectional bus the communication may either be half-duplex or full-duplex. Half-duplex implies that communication may take place in both directions but not at the same time while full-duplex implies simultaneous communication in both directions.

2.2

REDUNDANCY PRINCIPLES

To obtain the required survivability, the cross channel communication function has to be redundant so that no single failure will cause a system failure.

This redundancy may either be realized by transmitting over active redundant links or by using a primary active link backed up by passive spares. The latter implementation will require half-duplex or a full-duplex mechanization.

Only active redundancy will be acceptable for the ARCS because of the difficulty of proving that all processors in the system will be able to correctly switch to a spare upon any failure of a primary link, i. e., the difficulty of showing that no single point failure source exists.

2.3

TRANSMISSION MODES

Four different transmission modes will be considered, independent, sequential, synchronized and command/response. Independent transmission implies that the data exchange is accomplished without any interaction whatsoever between processors.

By sequential transmission it meant the scheduling in time between different processors such that transmission is performed in a prescribed sequence, each processor transmitting in a given time interval.

In synchronized transmission the processors exchange information on a word by word basis, transmitting simultaneously and storing the received data in a coordinated, synchronized manner.

The command/response mode is self-explanatory.

Of the four modes considered above, the command/response mode may directly be ruled out by the ARCS "noninterference" groundrule.

If synchronized transmission is to be used for ARCS, the timing reference for the data exchange has to be generated independently in each processor. This necessarily lowers the effective transmission rate, since the transmission intervals have to be expanded to accommodate time skew between channels. Synchronized transmission is therefore a less desirable alternative for the ARCS.

With regard to sequential transmission, the observation is first made that sequential transmission offers no advantages over independent transmission for unidirectional busses. Sequencing between processors requires scheduled busses. Second, the sequential mode requires scheduling, which has to be managed independently by each processor to satisfy the noninterference groundrule. Finally, sequential transmission greatly increases the time required for the data exchange if not compensated by higher transmission rates. The sequential mode therefore is also less desirable for the ARCS for the ARCS.

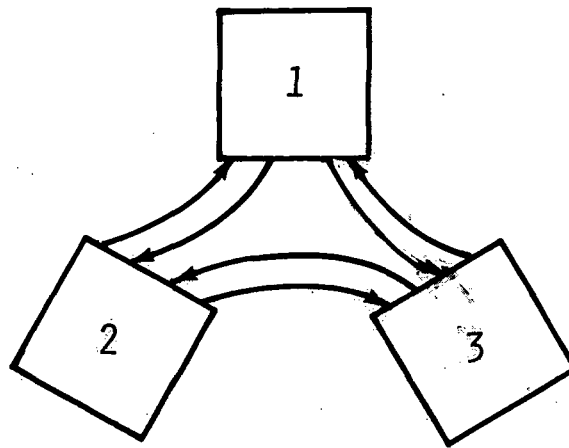
This leaves independent transmission as the only desirable transmission mode for the ARCS crosschannel communication.

PROCESSING TIME CONSTRAINT

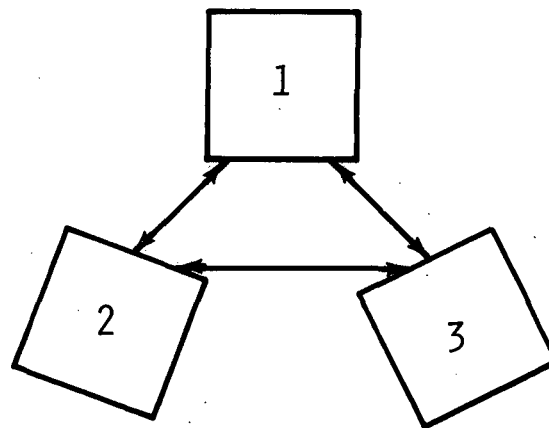
It is estimated that up to seventy-five percent of the total processing time will be required to handle the redundancy management and control law tasks for the ARCS. In a 20 ms frame, this leaves 5 ms for other tasks including crosschannel data transmission. The required amount of data exchanged per frame varies, but will be in the order of 400 words. The average processor execution time per transmitted word is therefore 12.5 μ s. From this, it is clear that the data exchange should take place with a minimum of processor involvement.

IMPLEMENTATION CONFIGURATIONS

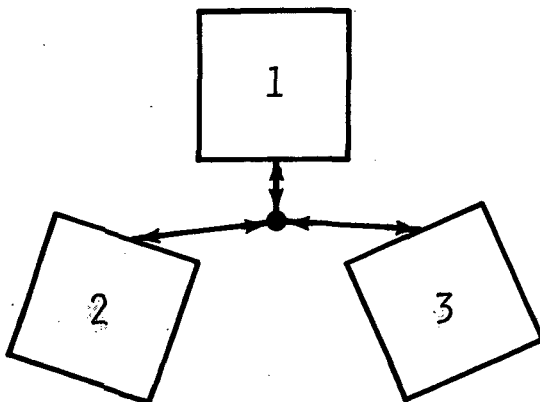
Figure I-1 shows three possible bus configurations. Configuration 1(a) uses six unidirectional, dedicated, transmission links. The delta configuration of figure I-1(b) consists of bidirectional busses connecting the three processors, and figure I-1(c) shows a "TEE" configuration using a



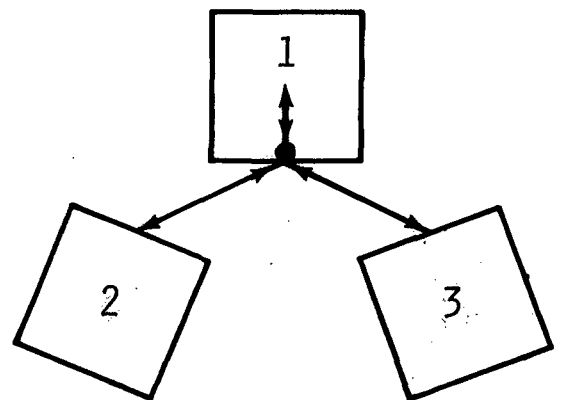
1(A), 6 - UNIDIRECTIONAL LINKS



1(B), DELTA



1(C), TEE



1(D), ALTERNATE TEE

FIGURE I-1 CROSS CHANNEL LINK CONFIGURATIONS

2.5

(cont'd)

common bidirectional bus. Figure 1-1(d), the "asymmetrical TEE" shows an alternate to the TEE which requires less cabling since the cable junction has been moved inside one processor interface.

2.6

SINGLE POINT FAILURE HAZARD AND RELIABILITY

A fundamental requirement for ARCS is that the system function should survive any single failure. Configuration 1(a) satisfies this requirement, since loss of one bus leaves the communication between two processors intact. A bus failure will therefore, at most, lead to redundancy degradation from triplex to duplex, provided the failure is properly localized by all three computers. The possibility of bypassing a faulty link by relaying over another computer is ruled out by transmission rate constraints and the ARCS redundancy management design.

A similar situation exists for the delta configuration where any bus failure will leave the communication between two computers intact. As in the previous case it is important that all computers are able to localize the fault.

A different situation exists for the TEE configuration where a single worst case failure at the bus junction conceivably could cause total loss of communication between computers. Redundancy is therefore required for this configuration. Figure 1-2 shows a possible alternative using a triplicated "alternate TEE" configuration.

With full-duplex implementation, the required fault tolerance could conceivably be obtained by two TEE's instead of three. A symmetric configuration is however required in order to satisfy the ARCS groundrule of identical computer hardware and software.

ORIGINAL PAGE IS
OF POOR QUALITY
523
253

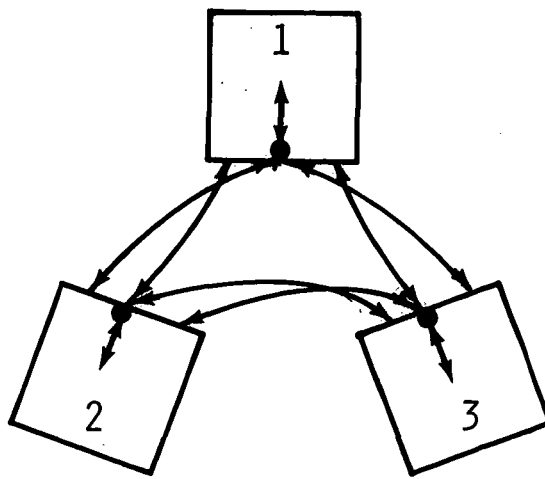


FIGURE I-2 TRIPLICATED TEE CONFIGURATION

Note that with unidirectional instead of bidirectional transmission the symmetric configuration of figure I-2 reduces to that of figure I-1(a). The bidirectional implementation exhibits higher fault tolerance but will be considerably more complex than the unidirectional. This increased complexity is hardly justifiable, since the resulting reduction in system failure probability only amounts to a few percent.

From the above considerations, we find that the configurations of figure I-1(a), using six unidirectional links, and the delta configuration of figure I-1(b), using three bidirectional full-duplex links, are the two remaining cross channel link contenders for the ARCS. However, the delta implementation will require somewhat more complicated hardware and will be more difficult to adapt to optical transmission, giving a slight edge to the configuration of figure I-1(a).

In conclusion, the most effective cross channel link implementation uses six unidirectional links. Each processor transmits data to the other two processors independently via two dedicated links.

3.0

DATA RATE AND TRANSMISSION FORMAT

There are several considerations in the selection of a data format for the cross channel communications link. Those to be considered here are:

- Data Word Size
- Data Rate
- Hardware Complexity
- Reliability of Communications
- Failure and Error Detection
- Existing Standards

3.1

DATA WORD SIZE AND DATA RATE

A data word will consist of 16 data bits, a 10-bit label and a parity bit for a total of 27 bits. If a data transfer rate of 35,000 words per second is required the bit rate will be 0.945×10^6 bits per second (bps) exclusive of overhead for bit, word and message synchronization. The baseline cross channel configuration requires a minimum of overhead for protocol, hence a bit rate of 2×10^6 bps would provide adequate margin and some growth potential.

Serial data transfer rates of 5×10^6 bps are well within the state-of-the-art so it is clearly feasible to use serial data transmission in this application and reap the associated benefits of simplicity, low cost and reliability.

3.2

TRANSMISSION FORMAT

Modern high-rate serial data transmission standards typically specify a self-clocking code transmitted on a single signal path per channel. This approach minimizes inter-connecting cables and avoids timing errors frequently associated with parallel clock, data and synchronization lines. Although MIL-STD-1553A pertains to a large-scale data bus system, the serial data transmission format specified therein provides a good model for the cross channel data format. This standard employs bipolar biphase-L encoded data in a word format that begins with a 3-bit synch pattern, 16 bits of data and a parity bit for a total of 20 bit-times per word. The synch pattern is used to obtain bit synchronization and to signify a data word or a command/status word. A message consists of a command or status word followed by up to 32 data words. This format has been found suitable for fiber-optic as well as wire data transmission for which it was designed.

A recommended data transmission format based on MIL-STD-1553A and adopted for the cross channel link requirements is as follows:

Sync Pattern: Command/Status and Data Sync (3 bits, in-valid code)
 Data Encoding: Biphase-L (Manchester)
 Word Format: Word Length 30 bits
 Sync (3 bits)
 Data & Label or Command/Status (26 bits)
 Odd Parity Bit (1 bit)
 Word Types: Command, Status, Data
 Message Format: Begin with Command or Status Word, followed
 by up to 64 data words and an optional status word
 containing a block check sequence.

This format should provide sufficient flexibility to accommodate a number of different communication protocols with a minimum of overhead. The 10 label bits which are not part of the MIL-STD-1553A format are to effect proper placement of data in a 1024 word RAM in the computer input section. The maximum length of one message will be 66 words of 30 bits each for a total of 1980 bits. Thus, a transfer of one block of data will require approximately 1 millisecond at a 2 megabit rate and 0.4 millisecond at a 5 megabit rate.

3.3

COMMUNICATIONS FAILURE AND ERROR DETECTION

The recommended communications format described above, together with a protocol suited to the selected cross channel link configuration, provides many ways to detect and isolate failures in the cross channel link. Message integrity checks that can be performed by hardware at each receiver are:

Valid Sync Patterns
 Valid Biphase-L Bits
 Valid Parity Bits
 Valid Address
 Correct Message Length
 Valid Check Sequence (optional)
 Valid Status Bits

The recommended message format provides command and status words that would allow the computers to pass instructions and failure status from one to another and to control the flow of traffic. A message comprised of a command word can request status or data and a reply would include a status word as a minimum plus any required data. Failure of a computer to respond to a command from a neighbor would signal the requestor that a failure has occurred.

Several error detection and error correction schemes have been developed for use in data communications systems. Selection of an error detection scheme for the cross channel link can be made on the basis of channel quality, consequence of an undetected error, added communications burden and cost. Probably the simplest and most widely used forward error detection scheme is parity. This scheme works reasonably well in serial data channels where bit errors are independent and the error rate is small. A parity check fails to detect errors that occur in pairs within the span of bits covered by parity. The main advantage of parity is its simplicity of implementation and low cost.

A more sophisticated and effective error detection scheme is the cyclic redundancy check (CRC). In this scheme a number of check bits (a code) is computed as a group of data bits is being sent, and the check bits are sent following the data. At the receiver a similar computation is performed on the received data and any disagreement of the receiver derived check bits with those sent following the message signals an error in the message. This check can detect all errors within a span of bits equal to the length of the code and it can also detect a large percentage of errors scattered throughout the data and exceeding the span of the code. It can be shown that a 16 bit CRC protecting 64 words of 26 bits each can yield an undetected bit-error rate at least 1000 times smaller than would be achieved with a parity check on each word. The CRC is easy to implement because the necessary computation at each terminal can be performed by a circuit in a single 14-pin package. The main disadvantage to this scheme is the delay of the integrity check until the end of a data block transfer. Thus, the data block must usually be held temporarily at the receiver until the check can be completed. A FIFO memory could perform this function but is an added cost.

In the cross channel link the communication channel quality should be such that the raw bit error rate will be completely negligible. Thus, the function of the data integrity check will be to defeat errors caused by equipment failures or abnormal interferences such as electrical transients. These error sources are likely to create burst errors which may defeat a simple parity

check. Hence, a 16-bit CRC check for each block transfer of 64 words or less is recommended. We also recommend word parity as a backup to the CRC check for detection of isolated, widely separated bit errors. These two schemes working together will form a powerful error detection system. A forward error correction scheme that would add somewhat more complexity is not necessary in this application because a detected error could initiate a request to repeat the message or the defective data block can simply be discarded.

4.0

OPTICAL VS ELECTRICAL COMMUNICATIONS

In the previous sections a comparison was made between various configurations without regard to the transmission medium being used. In this section the advantages of optical vs electrical cross channel communications will be explored. Discussions of wire and fiber optic implementation of the baseline configurations are also presented.

4.1

ADVANTAGES OF FIBER OPTIC DATA TRANSMISSION

The transmission of digital signals over fiber optic lines has some unique features which make it an attractive alternative to conventional wire techniques for cross-channel data communications. Some advantages of fiber optic data transmission are:

- Large Bandwidth
- EMI/EMP Immunity
- No Cross Talk
- No Electrical Short Circuits
- No Ground Loop Problems
- Electrical Isolation of Terminals
- Light Weight
- Graceful Connector Degradation

The reliability of a fiber optic data link, from a signal integrity viewpoint, is very high due to its immunity to electromagnetic interference (EMI), electromagnetic pulse (EMP) and cross talk. This reliability is enhanced by the fact that the optical channel is a non-conductor of electricity; thus, there is no possibility of data link ground loops or electrical short circuits. The electrical isolation of transmitter and receiver precludes the propagation

4.1

(cont'd)

of an electrical fault from one processor to another via the cross channel data link. Furthermore, fiber optic connectors tend to degrade gracefully, reducing the chance of intermittent connections sometimes experienced with wire system connectors.

The data rate of a fiber optic data link implemented with off-the-shelf LED's is limited by the speed of the light emitting diode. Typical LED rise and fall times are 10-15 nsec. Experimental LED's are available with rise times as low as 1 nsec. Therefore, data rates higher than 65 Mbits/sec are possible with presently available components. This is far greater than the data rates possible with twisted shielded pair wire cross channel data links.

The mechanical ruggedness of fiber-optic data links is currently being investigated. Some preliminary test results suggest that the mechanical reliability of fiber optic cable is greater than wire. Furthermore, new fibers and protective jacketing materials are being developed which should provide uncontested mechanical integrity.

4.2

IMPLEMENTATION OF THE BASELINE CROSS CHANNEL CONFIGURATION

The baseline cross channel configuration, shown in figure I-1(a), employs 6 unidirectional data links between the 3 processors. Each processor has two transmitters and two receivers. The relative complexity of implementation with wire and fiber optics is comparable. With the short distances between computers and a 2 megabit per second or lower data rate, either system can be implemented without difficulty.

To obtain electrical isolation between computers in the wire system, it is necessary to use transformer coupling in the transmitter and/or receivers. Although the desired isolation can be verified at the time of manufacture, it is difficult to measure, hence insure, the isolation with the equipment in service. With a fiber-optic system, however, complete electrical isolation between terminals is inherent in the transmission medium and electrical fault propagation from one processor to another via the data link is not possible.

The relative cost of wire and fiber optic data transmission systems is a function of both relative complexity and state-of-the-art. Wire data transmission systems are highly developed whereas their fiber-optic counterparts have only recently emerged as contenders. Thus, it is to be expected that for systems of equal complexity, a wire implementation would be lower in cost at this time. Within a few years, however, fiber optic data transmission systems should be fully cost competitive with wire and a selection would be based largely on other factors.

The relative complexity of wire and fiber optic implementation for the ARCS cross channel link can be assessed with the aid of Figure I-3.

The computer I/O, encoder and decoder are virtually identical for either implementation, thus, any basic difference in complexity or cost will be found in the transmitter, receiver and transmission medium. Wire and fiber optic transmitters are both quite simple and comparable in complexity. A fiber optic receiver for a short data link or small data bus is no more complex than a wire line-receiver at present, because either can be purchased as single-package integrated circuits. The wire line-receiver is less costly at present because of product maturity but this is probably a temporary situation. Finally, as a transmission medium, optical fibers and terminations are currently more costly than wire but promise a cost advantage in the future because of simplicity and low-cost raw materials. However, because of the small amount of cable required to interconnect the computers, small differences in cable costs will have little effect on overall system costs.

In conclusion, a wire cross channel link has a cost advantage at present due to maturity of the technology. The cost difference is difficult to assess on a quantitative basis because of the rapidly changing state of fiber optic technology. All indications point to increased use and reduced cost of fiber optics in the near future and ultimately the choice will rest almost entirely on the system performance and failure effects trades.

541532

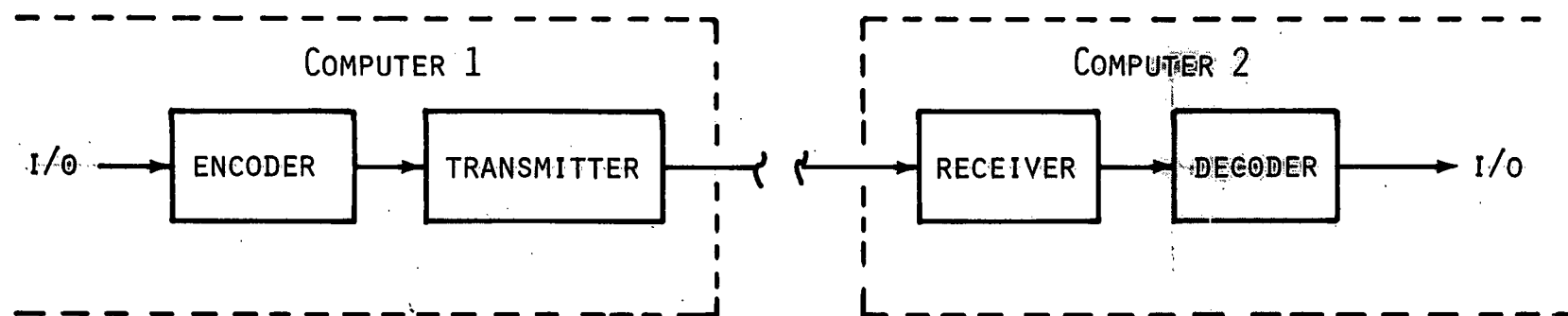


FIGURE I- 3 CROSS CHANNEL DATA LINK HARDWARE - BLOCK DIAGRAM

SUMMARY

A cross channel communication link implementation, which uses six unidirectional links between the three processors, is recommended for the ARCS. Each processor transmits data to the other two processors via two dedicated links in an independent manner. This implies that each processor has the ability to store data directly into a receiving buffer without interfering with the function of the receiving processor.

The configuration suggested, which coincides with the ARCS baseline cross channel link configuration, is well suited for adaption to optical implementation. The cost of optical transmission is presently higher than for electrical but this difference is insignificant in relation to the total system cost and is expected to decrease in the future.

REFERENCES

1. Morris, Everett W.: "Stumbling Blocks Can Be Avoided When Seeking Airworthiness Approval of Digital Flight Control Systems." AIAA paper 75-576, April 2-4, 1975.
2. "Airworthiness Requirements for Automatic Landing--Including Automatic Landing in Restricted Visibility Down to Category III." BCAR paper No. 367, prepared by Air Registration Board, Section D, Airplanes, January 1970.
3. "Environmental Conditions and Test Procedures for Airborne Electronic/Electrical Equipment and Instruments." Document DO-160, prepared by Radio Technical Commission for Aeronautics SC-112, February 1975.